République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique.

Université Saad Dahlab, Blida
USDB.

Faculté des sciences.
Département informatique .

Mémoire pour l'obtention
d'un diplôme d'ingénieur d'état en informatique.
Option : Intelligence artificielle

Sujet :

Optimisation du processus de fabrication par simulation de l'outil de production intégrant le couplet « modèle géométrique pièce/ modèle cinématique de MOCN »

Présenté par :

Loudjedi Fatma Zohra
Zahra Fatma Zohra

Proposé par :
M^me^.OUKID khouas
Mr. BENDIFFELLAH. H
M^elle^. MAAZOUZI. M

Organisme d'accueil : Centre du Développement des Technologies Avancées

Année :2004-2005

Dédicaces

A la mémoire de ma grande_mère. qui souhaitait toujours de me voir un jour devenir ingénieur. Tu me manques beaucoup .

A mon père , Merci de me guider et de m'orienter dans la bonne voie,. Merci de m'avoir aidé de surmonter les difficultés .Merci pour ton amour ,tu es mon exemple dans la vie .

A ma mère , Merci pour ton amour , tes encouragements et ta compréhension , je n'arrive jamais de te donner ce que tu mérites

je suis très fière d'être votre fille ,

A mes deux frères,Yagoub et Ibrahim, vous avez été d'un soutien inestimable

A mes deux sœurs,,Zakia et Meriem, vous avez été mon soutien dans les moments difficiles.

A mon binôme Ratiba , j'ai passé avec toi une agréable année

A toute ma famille.

A mes amies : Wided , Fatma, Samira, Hadjira, Samia, Djahida, Razika , Fatima , Salima , Fatma Zohra, Ibtissem, Fatiha, Amel, Lydia, Souad, Moufida, Nadia…

Je vous dédie ce travail

LOUDJEDI Fatma Zohra

Dédicaces

A ma mère , l'être le plus important sur cette terre pour moi, ton amour, ta charité l'essence de mon existence . Merci pour tout ce que tu as fait et ce que tu fais pour moi.

A mon père , merci pour ta compréhension , tes encouragements, ton aide , et ton soutien pendant toutes ces longues années de ma existence.

A mon grand frère Rachid qui m'a aidé à mettre le premier pas sur le long chemin de savoir.

A mon petit frère , Redouane , merci d'être non seulement un frère mais aussi un ami.

A ma nièce Hayet , mes neveux Issam, et le petit Mohamed que j'adore

A mon binôme , Fatima , on a passé ensemble une très heureuse années . Merci pour ta compréhension, et ta gentillesse.

A toute ma famille.

A mes amies Amel , Nawel, Heifaa, Ibtissem, Fatiha, Amel, Lydia, Moufida , Nadia...

Je vous dédie ce travail

ZAHRA Fatma Zohra

Remerciements

Louange à notre Seigneur "ALLAH" qui nous a doté de la merveilleuse faculté de raisonnement. Louange à notre Créateur qui nous a incité à acquérir le savoir. C'est a Lui que nous adressons toute notre gratitude.

Nous remercions vivement les membres du jury de nous avoir fait l'honneur d'être rapporteurs et examinateurs tout en apportant leurs remarques et leurs contributions à l'élaboration de ce mémoire.

Tous nos remerciements aux personnes qui, par leurs conseils et leurs encouragements ont contribué à l'aboutissement de ce travail :

A Mr Bendiffellah Hassen , ingénieur mécanicien au LRIA du CDTA , de nous avoir proposé ce sujet et qui nous a fait bénéficier beaucoup de son expérience et qui a tout fait pour faire aboutir ce travail.

A Melle Maazouzi Meriem , attaché de recherche au laboratoire de robotique & d'intelligence artificielle LR&IA pour son soutien moral et ses encouragements.

A Mme Oukid-Khouas d'avoir suivi et critiqué ce travail .

A Mr Bey Mohamed pour son aide qu'il a apporté pour la réalisation de ce travail.

A Mme Boumehdi Djaouida pour son aide .

Nous exprimons notre profonde reconnaissance à tous ceux qui nous ont aidé, de prés ou de loin, matériellement ou moralement, partager nos efforts pour voir naître ce modeste ouvrage.

Nous voudrions aussi exprimer notre profonde gratitude à nos parents pour tout l'amour qu'ils n'ont pas cessés de nous donner.

RESUME :

Cette étude porte sur l'optimisation des processus de fabrication, l'une des facettes de cette optimisation est l'amélioration continue des gammes de fabrication par leur validation virtuelle . Cette vérification est à niveau hiérarchique élevé, implique une combinaison de couples (modèle géométrique pièce/ configuration machines_outils).

Les surfaces de formes libres dites gauches sont difficilement usinables. Ces surfaces sont généralement usinables sur des MOCN.

L'approche proposée consiste à vérifier l'usinabilité des surfaces gauches sur les MOCN en analysant leur propriétés géométriques et les propriétés cinématiques des différentes configuration des MOCN.

L'apport majeur de cette modélisation , est d'associer une pièce de forme gauche à un type de machine outil, et l'intégration de l'aspect dynamique , par définition de l 'activité cinématique.

ABSTRACT :

This study relates to the optimization of the manufacturing processes , one of the facets of the optimization is the continu improvement of the manufacturing by their virtual validation. This checking implies a combination of couple (geometrical model piece / configuration of machines tools).

Sculptured surfaces , or so-called free form surfaces , are difficultly machined. Those are usually produced by CAD/CAM and numerical controlled (NC) machine.

The approach proposed consists in checking of the machinability of sculptured surfaces on the numerical controlled milling machines by the analyze of their geometrical properties and the Kinematic's properties of the different configuration of the MOCN.

The major contribution of this modelling , is to associate a sculptured surface objects to a type of machine tools .

Sommaire

Liste des figures

INTRODUCTION GENERALE

DANS CETTE PARTIE

- SITUATION DU PROBLEME
- L'OBJECTIF DU TRAVAIL
- DESCRIPTION DU TRAVAIL

I- SITUATION DU PROBLEME

L'évolution très rapide de la technologie a fait en sorte que le marché industriel est devenu de plus en plus compétitif et exigent.

Pour prétendre rester compétitifs, l'amélioration continue des performances de systèmes de production est devenue obligatoire. Il est important de réduire le temps et le coût de fabrication tout en améliorant la qualité du produit et les conditions de travail, en offrant aux utilisateurs des ressources intégrées réduisant les tâches itératives et coûteuses.

Le développement de modèles robustes des processus de fabrication suscite une attention croissante des praticiens industriels et des chercheurs .Cette tendance est liée aux besoins d'outils numériques et analytiques capables de prédire les performances des processus de fabrication afin de concevoir simultanément le produit et son processus d'industrialisation.

Ces nouveaux outils doivent s'intégrer dans les environnements de CFAO, et dans le concept plus général du « Virtual Manufacturing ». Or la pratique industrielle dans ce domaine reste encore fortement fondée sur l'expérience, le savoir-faire et les expérimentations.

C'est pourquoi, et dans le cadre du développement d'outils de conception et de fabrication des surfaces irrégulières (surfaces gauches) initié par l'équipe de CFAO de la division robotique et productique du centre de développement des technologies avancées (CDTA), nous envisageons de concevoir et d'implémenter une application sous environnement graphique pour la validation virtuelle de l'usinabilité des surfaces gauches.

On peut synthétiser l'objectif global de notre travail de la façon suivante :

' Optimisation du processus de fabrication par simulation de l'outil de production, intégrant le couplet « modèle géométrique pièce/modèle cinématique de MOCN »'

Traditionnellement, le processus de fabrication des formes libres, dites formes gauches, est basé sur des usinages successifs. Ces formes (surfaces) utilisées généralement dans des secteurs industriels tel que l'aéronautique, la construction navale, l'aérospatiale ainsi qu'une grande variété de produits modernes tel que les téléphones cellulaires, les moules... sont difficilement usinables.

Les surfaces complexes (gauches) sont définies par des équations paramétriques à coefficients vectoriels. Elles ne peuvent pas être usinées sur des machines conventionnelles à cause d'une part de la nature complexe du processus de fabrication, et d'autre part de l'hétérogénéité des informations relatives à la géométrie de ces surfaces.

Pour cela, on utilise des **M**achines **O**util à **C**ommande **N**umérique (**MOCN**), plus particulièrement des fraiseuses numériques.

La validation virtuelle, ou bien la simulation de l'usinage, comme un moyen d'essai et de vérification de l'usinabilité des pièces de forme complexe est devenue une partie importante du processus de fabrication moderne.

II- L'OBJECTIF DU TRAVAIL

Pour mieux cerner le problème cité précédemment, nous devons :

1) En premier lieu, développer une application logicielle permettant de modéliser les machines outils selon le modèle géométrique et le modèle cinématique.

2) En deuxième lieu, développer une application permettant de vérifier la machinabilité des pièces de forme libre.

3) En dernier lieu, optimiser l'usinage des formes gauches en fonction des configurations des MOCN permettant d'obtenir une surface donnée. Pour ce faire, une étude sur les contraintes liées à la machinabilité des surfaces gauches est à faire.

Le résultat est un système qui, à partir d'une pièce de forme libre, capable d'associer un type de machine outil tout en définissant la cinématique minimale de la machine permettant d'obtenir cette surface.

L'application est à développer sous Windows en utilisant le ***BUILDER C++*** et ***OpenGL***.

III- DESCRIPTION DU TRAVAIL :

Le présent mémoire tourne autour de cinq chapitres.

Nous commencerons d'abord par une introduction générale dans laquelle nous présenterons la problématique du sujet.

Le premier chapitre sera consacré à l'étude de la modélisation géométrique des courbes et des surfaces utilisées dans la représentation des formes complexes.

Dans le deuxième chapitre, nous développerons une étude sur les machines outils à commande numérique et plus particulièrement les fraiseuses numériques.

Dans le chapitre 3, nous étudierons l'usinage des surfaces gauches sur les fraiseuses numériques.

Dans le chapitre 4 nous présenterons la conception adoptée pour la réalisation de ce travail.

Le chapitre 5 sera consacré aux tests et à la validation.

Nous terminerons par un bilan du travail réalisé et les éventuels perspectives et améliorations à donner à ce travail.

1

LES COURBES ET LES SURFACES GAUCHES

DANS CE CHAPITRE

- DESCRIPTION SUCCINCTE D'UNE CHAINE CFAO
- MODELISATION DES COURBES IRREGULIERES
- MODELISATION DES SURFACES IRREGULIERES

I- INTRODUCTION

Les surfaces complexes sont très utilisées dans la conception des modèles des pièces moulées, les carrosseries des voitures, les coques des navires et dans l'aviation. Ce large champ d'application des surfaces, nécessite des outils et des méthodes adéquats qui facilitent la conception, l'édition et la modification des surfaces d'une manière interactive. Pour cela, les modèles des courbes et des surfaces doivent avoir des interprétations géométriques et permettent de connaître intuitivement la forme finale de la courbe ou de la surface .

Dans ce chapitre , nous allons donner en premier lieu une description succincte d'une chaîne CFAO. Ensuite , nous allons présenter la modélisation des courbes irrégulières, et nous terminerons en dernier lieu par les méthodes de conception des surfaces irrégulières.

II- DESCRIPTION SUCCINCTE D'UNE CHAINE CFAO

La CFAO est une technique relativement récente introduite dans l'industrie en raison de la grande flexibilité qu'elle offre dans le domaine de la production ainsi que pour ses performances au niveau de la qualité , la quantité , le coût et les délais de fabrication . Elle est constituée généralement des systèmes suivants:

II.1- La Conception Assistée par Ordinateur CAO [1] :

Le système de *CAO* est composé d'un ensemble d'outils informatiques (matériels + logiciels) devant permettre le dialogue entre l'utilisateur et le système.
La CAO regroupe quatre activités qui ont pour point commun, la conception des pièces et des systèmes : *modélisation géométrique, l'analyse, le contrôle et le dessin.*

Dans le cas de *modélisation géométrique*, on génère des images géométriques sur un écran vidéo, à partir de description mathématique d'objets stockées dans la mémoire d'un ordinateur.

Le système de CAO permet d'afficher le modèle géométrique de la pièce sur l'écran graphique interactif du poste de travail et lui faire subir, si nécessaire, un certain nombre de transformations, notamment des rotations.

Le programmeur indique alors, au système la nature des matériaux et les surépaisseurs à prévoir entre le profil brut et la pièce finie, il peut de la même façon, concevoir les outillages et les dispositifs de fixations de pièces tout en les visualisant.

Le système de CAO est constitué de plusieurs modules. Parmi lesquels on peut citer : le module de repérage des surfaces ,le module extracteur de caractéristiques, le module de conception par caractéristiques de contraintes dimensionnelles et fonctionnelles....

La CAO permet :

- La création automatique des perspectives.
- La réalisation d'éclatés.
- La modélisation en 2 Dimensions et 3 Dimensions.
- La réalisation de calculs complexes et longs (Ex : calcul de résistance de charpentes).

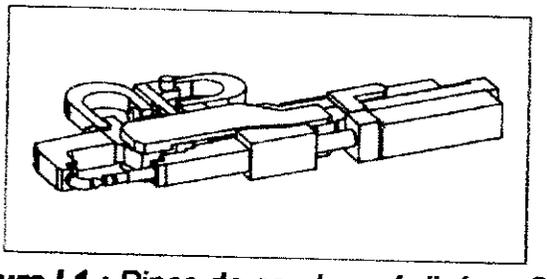


***Figure I.1** : Pince de soudure réalisé en CAO*

II.2- La Planification Assistée par Ordinateur (PPAO) : [1]

Il s'agit de déterminer automatiquement, la gamme d'usinage optimale établie à partir des informations issues du module de CAO.
Cette gamme est une suite logique des différentes étapes de réalisation de la pièce, en fonction des moyens disponibles et des résultats à obtenir.
En d'autres termes, c'est une séquence d'opérations d'usinage contenant le choix du montage, les outils de coupe et la condition de coupe pour chaque opération ainsi que les règles d'antécédences entre opérations.

La PPAO permet :

- de réaliser la gamme opératoire (liste des opératoires décrivant le processus de fabrication).

II.3- La Fabrication Assistée par Ordinateur :

La FAO désigne dans son sens le plus large, l'utilisation d'un ordinateur dans les lieux de production et de fabrication pour la planification des processus, le contrôle de la qualité, la technologie de groupe ou, encore, la planification des besoins en composants ....

L'utilisation de logiciels de FAO pour générer les programmes de commandes numériques d'usinage des surfaces simples ou complexes est devenue très courante.

Le logiciel doit être capable à partir d'un modèle CAO de la pièce, de générer les trajets de l'outil qui permettent d'usiner la pièce, et éventuellement de transformer ces trajets en un programme compréhensible par le directeur de commande numérique. **[1]**

Générer des trajets outil, c'est trouver les positions de l'outil successives qui permettent d'usiner une surface selon des trajectoires définies par le programmeur machine outil. Les trajectoires sont en général des courbes 3D, intersections entre les surfaces à usiner et les surfaces de guidage de l'outil. La stratégie d'usinage définit les surfaces de guidage qui sont le plus souvent des plans parallèles entre eux.

Un trajet est représenté par une succession de couples formés par la position de l'extrémité de l'outil et la direction de l'axe outil. Comme l'encombrement de l'outil n'est pas pris en compte dans cette représentation, l'encombrement doit être étudié lors du calcul des positions outil sinon des collisions entre l'outil et la surface risquent de se produire. Par exemple, si le rayon de courbure d'une partie concave de la surface est plus petit que le rayon de l'outil, alors il y a interférence entre l'outil et la surface.

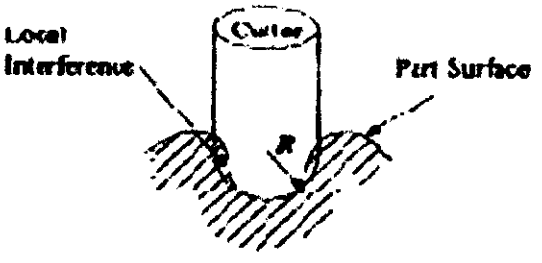


***Figure I.2** : Interférence entre l'outil et la surface*

La FAO permet :

- De simuler le parcours d'outils avant l'usinage
- De réaliser des parcours d'outils.
- D'augmenter la productivité du couple opérateur/machine en permettant à celui-ci de préparer les travaux suivants pendant l'usinage.

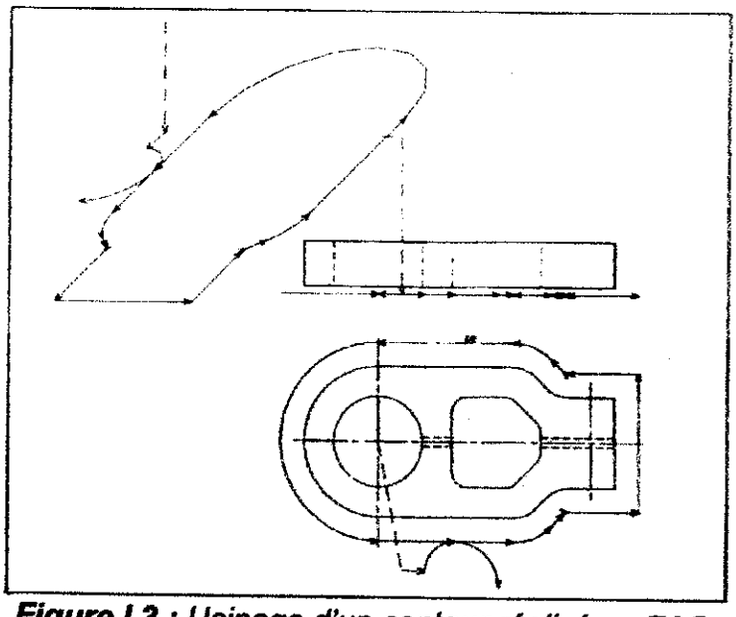


Figure I.3 : Usinage d'un contour réalisé en FAO

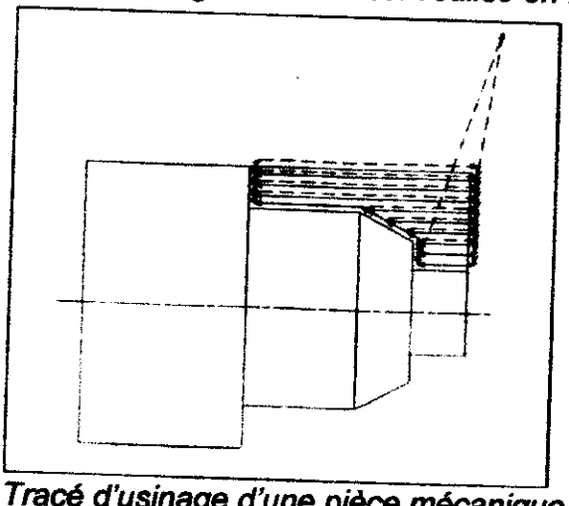


Figure I.4 : Tracé d'usinage d'une pièce mécanique (tournage)

III- LA MODELISATION DES COURBES IRREGULIERES [1]

Tous les objets réels, et mêmes les objets techniques issus de la CAO, ont des formes douces, par rapport aux formes anguleuses obtenues à partir de segments de droite [polygon mesh].

L'utilisation de courbes apporte des avantages importants pour le traçage et surtout pour la modélisation des objets :

- La courbe est stockée de manière plus efficace en mémoire.
- Son traçage est plus précis (pas d'effet de cassure).
- Il n'est pas nécessaire d'interpoler entre les points.
- La modification de l'allure de la courbe est aisée.

III.1 - Définition d'une courbe irrégulière :[1]

Une courbe irrégulière peut être construite à partir de :

- Lissage des points.
- Intersection de deux surfaces.
- Courbe parallèle à une autre courbe à une distance fixe.
- Raccordement de deux courbes ...etc

III.2- Méthodes de modélisation des courbes irrégulières :

Trois principaux types de méthodes numériques sont utilisés pour la modélisation par courbes :

• ***Le lissage :***

Le lissage génère une courbe à partir d'un nuage de points fournis par l'utilisateur. La fonction générée ne passe pas obligatoirement par tous les points fournis.

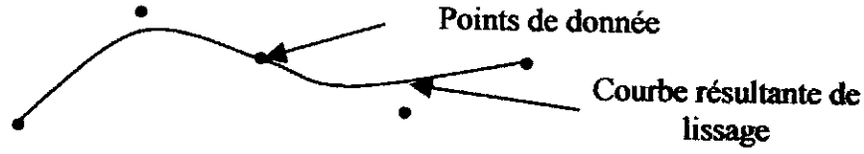


Figure I.5 : *Exemple de lissage*

• ***L'interpolation :***

L'interpolation génère une courbe qui passe par tous les points fournis par l'utilisateur.

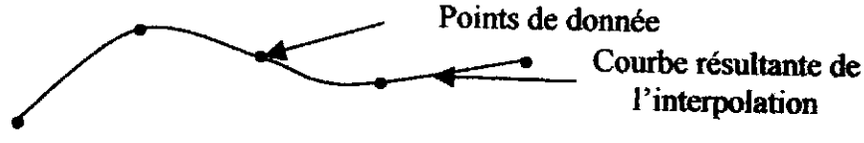


Figure I.6 : *Interpolation d'un nuage de points pour une courbe*

• ***Les méthodes à pôles ou à points de contrôles :***

Ces méthodes permettent un ajustement de la géométrie à réaliser à partir des points de contrôle. La courbe générée passe par le premier et le dernier point fourni.

III.3- Types des courbes irrégulières :[1]

Les courbes utilisant les méthodes précédentes sont détaillées dans ce qui suit :

- **Les courbes de Bezier :**

Les courbes de Bezier ont été développées pour modéliser les profils tout en ayant la possibilité de modifier facilement et itérativement la géométrie pendant la conception.

Ainsi, la notion de « pôle » ou de « points de contrôle » a été utilisée pour altérer l'allure de la courbe. Cette notion a été introduite pour offrir à l'utilisateur la possibilité d'agir sur la forme de la courbe.

Un polygone de contrôle est défini par un ensemble de sommets ordonnés de *n+1* points $S_0, S_1, \ldots\ldots S_n$. pour les *n+1* points de contrôle, le polynôme interpolé est de degré « n ».

L'équation de la courbe de BEZIER associé à un polygone de contrôle S0, S1, ..,Sn est :

$$P(u)= \sum_{i=0}^{n} B_{i,n}(u) \bullet S_i \qquad S_i= \begin{pmatrix} Xi \\ Yi \\ Zi \end{pmatrix} \qquad u\in[0,1]$$

Où: S_i : vecteur position des points S0,S1,....,Sn

n : degré de la courbe.

$B_{i,n}(u) = C_i^n \bullet u_i \bullet (1-u)^{n-1}$: les coefficients du polynômes de Bernstein.

$$C_i^n = \frac{n!}{(n-i)!i!}$$

- **Les courbes B-Spline:**

Les courbes de Bézier présentées précédemment ont deux principales limites :

- Le degré de polynôme est fonction du nombre de points.
- Le changement de position d'un point de contrôle influence l'allure de toute la courbe (contrôle global).

La modélisation par la fonction B-Spline permet de remédier à ces inconvénients. Ainsi, en plus de la conservation des propriétés de Bézier les fonctions B-Spline permettent :

- Une altération du degré de polynôme
- La modification locale (contrôle local).
- L'obtention d'une courbe différente pour les mêmes points de contrôle en modifiant le degré de polynôme

Une courbe B-Spline est une forme paramétrique polynomiale. C'est-à-dire qu'elle est construite à partir d'une succession de polynôme de degré *n*, se raccordant entre eux a des abscisses u_i de telle façon que la continuité en ces points soit assurée jusqu'à la dérivée *n-1* au maximum.

Les abscisses u_i des points de raccordement seront appelées *des nœuds* et l'ensemble des nœuds associés à une courbe B-Spline sera nommée *séquence nodale*.

On appelle *intervalle*, le domaine de variation du paramètre *u* entre deux nœuds successifs. Les nœuds et les séquences nodales contribuent directement à la définition d'une courbe B-Spline avec une importance égale à celle du polygone de contrôle.
La courbe B-Spline de degré *m* est définie par *n+1* pôles $S_0, S_1 \ldots S_n$ (les sommets du polygone de contrôle) et des fonctions d'interpolations (les fonctions B-Spline) de degré m et avec une séquence nodale $U_0, U_1, \ldots, U_{n+m+1}$.

La formulation mathématique des courbes B-Spline pour les *n+1* points de contrôle :

$$\mathbf{P(u)} = \sum_{i=0}^{n} \mathbf{N_{i,m}(u)} \bullet \mathbf{S_i}$$

avec :

S_i : le vecteur de position.
m-1 : l'ordre du polynôme de la fonction B-Spline.
$N_{i,n}$: les coefficients du polynôme de base définis récursivement par :

$$Ni,1(u) = \begin{cases} 1 & \text{si} \quad u_i <= u <= u_{i+1} \\ 0 & \text{autrement} \end{cases}$$

$$N_{i,m}(u) = \frac{u-u_i}{U_{i+m+1}-u_i} \bullet N_{i,m-1}(u) + \frac{u_{i+m-u}}{u_{i+n}-u_{i+1}} \bullet N_{i+1,m-1}(u)$$

où la convention suivante est assumée 0/0=0

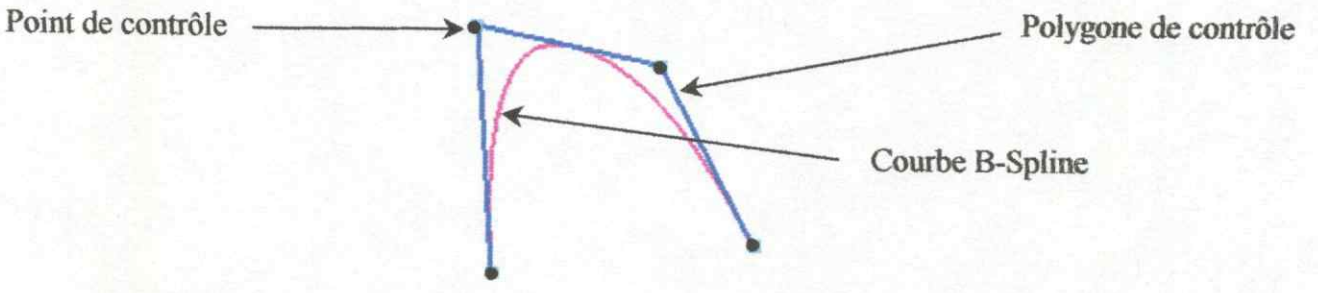


***Figure I.7** : Courbe B-Spline et son polygone de contrôle*

- La seule donnée du polygone de contrôle définit entièrement la courbe.
- La courbe est tangente au premier segment S_0S_1 en S_0 est au dernier segment S_{n-1} Sn en S_n.
- Une courbe définie par $S_0, S_1, \ldots, S_n$ reste identique si elle est définie par $S_n, S_{n-1}, \ldots S_0$
- La courbe passe par les deux points extrêmes du polygone de contrôle.

- **Les NURBS** :

Le terme NURBS signifie ***Non-Uniform Rational B-Spline*** (B-Spline rationnelle non uniforme). Plus précisément :

- ***Non uniforme*** signifie que les nœuds peuvent ne pas être équidistants. Cela peut être utile pour la modélisation des surfaces irrégulières.

- ***Rationnelle*** signifie que l'équation représentant la courbe ou la surface est exprimée sous la forme d'un rapport entre deux polynômes, et non sous la forme polynomiale comme pour le cas des B-Spline. L'équation rationnelle permet d'obtenir un meilleur modèle pour certaines courbes et surfaces importantes, en particulier pour les sections coniques, (les cônes, les sphères,... etc.).
- **Une *B-Spline*** correspond à une *Spline de base*.

De même que les autres modèles, la principale utilisation du modèle des NURBS concerne la représentation des coniques et l'accumulation qu'il réalise entre les propriétés de ces courbes et des courbes B-Spline.

Parmi les avantages des courbes NURBS, la possibilité de représenter des courbes dont la répartition des points donnés n'est pas du tout régulière.

IV- MODELISATION DES SURFACES IRREGULIERES

La conception et la modélisation des surfaces sont très importantes dans le domaine de la conception assistée par ordinateur (CAO). Les concepteurs de surfaces ont besoin d'un système de modélisation qui leur permet de concevoir et de manipuler les surfaces d'une manière interactive.

IV.1- Définition des surfaces gauches (irrégulières) :

Du point de vue de l'obtention de la forme, ce sont des surfaces dont la réalisation ne peut pas être obtenue par un mouvement simple de l'outil.

Du point de vue de la conception , elles ne peuvent pas être définies par des équations mathématiques simples ,elle sont définies par des équations paramétriques à coefficients vectoriels.

IV.2- Les méthodes de modélisation des surfaces irrégulières :

Les méthodes qui peuvent aider le concepteur sont classées en :

IV.2.1- Méthodes basées sur les points :

- **Interpolation de la surface :[4][5]**

La surface résultante de l'interpolation doit passer par l'ensemble des points fournis. Le traitement effectué sur ces points conduit à une modification interactive de la forme de la surface.

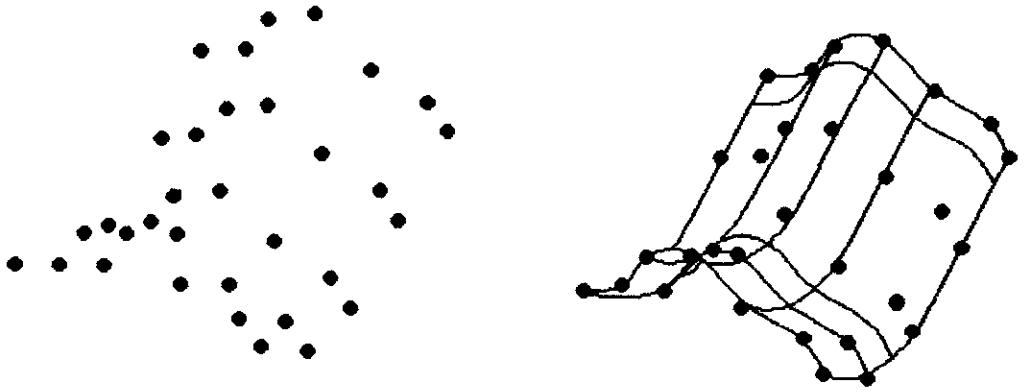


***Figure 1.8** : interpolation d'une surface*

Cette interpolation peut être faite d'une manière globale ou d'une manière locale :

➢ **Interpolation globale:[2]**

Dans l'interpolation globale, tous les points sont utilisés lors de la construction de la surface, ce qui rend chaque point sur la surface résultante, affecté par tous les autres points et la modification d'un point change la forme de la surface toute entière.

➢ **Interpolation locale :[2]**

Contrairement à l'interpolation globale, l'interpolation locale est appliquée après avoir devisé l'ensemble des points en sous-ensembles. Chaque sous ensemble définit un morceau de la surface. L'interpolation est faite pour chaque sous-ensemble et les surfaces générées sont jointes entre elles avec des contraintes de continuité pour générer la surface souhaitée.

Avec l'interpolation locale, le changement d'un point ne change que le voisinage du morceau de la surface dont il appartient.

- **Approximation de la surface : [4][3]**

Contrairement à l'interpolation, l'approximation est appliquée pour déterminer une surface approchée aux données fournies (les points de données)(figure 2). Généralement, elle est utilisée dans le cas où les valeurs d'entrée sont des valeurs approchées (mesurées).

Pour avoir un meilleur contrôle de la forme de la surface, on associe à chaque point un poids qui détermine l'approche ou l'éloignement de la surface de ce point.

Comme pour l'interpolation, l'approximation peut se faire d'une manière globale ou d'une manière locale.

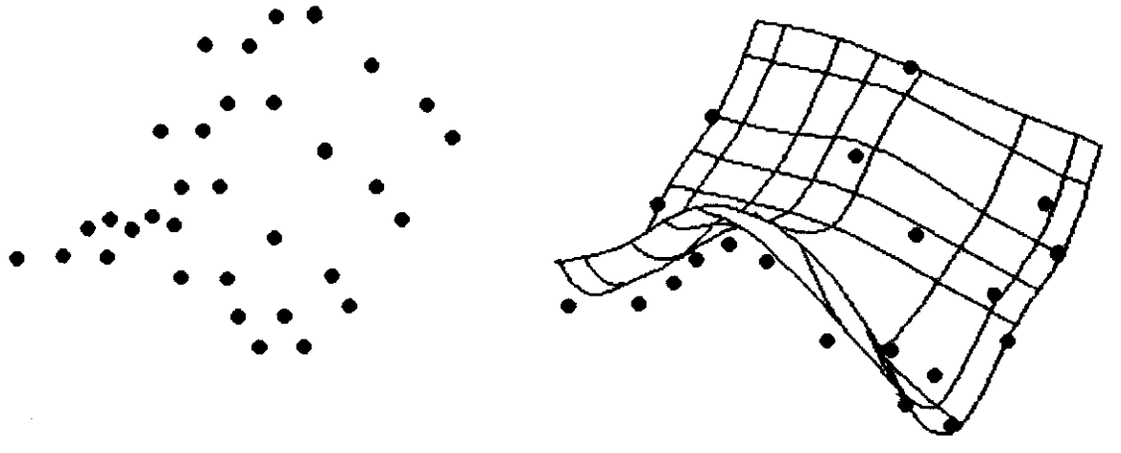


***Figure I.9** : Approximation d'une surface*

- **Surface de Bézier :[4]**

La surface de Bézier est définie par un réseau de (m+1)×(n+1) points de contrôle $P_{i,j}$, où $0 \le i \le m$ et $0 \le j \le n$.

L'équation de la surface de Bézier de degré (m,n) définie par $m+1$ lignes et $n+1$ colonnes de points de contrôle est donnée par :

$$P(u,v) = \sum_{I=0}^{m} \sum_{j=0}^{n} B_{m,i}(u) B_{n,j}(v) P_{i,j}.$$

Où : $B_{m,i}(u)$ et $B_{n,j}(v)$ sont les fonctions base Bézier dans les directions u et v respectivement . Ces fonctions sont définies par :

$$B_{m,i}(u) = \frac{m!}{i!\,(m-i)!} u^i (1-u)^{m-i}$$

$$B_{n,j}(v) = \frac{n!}{j!\,(n-j)!} v^j (1-v)^{n-j}$$

Puisque $B_{m,i}(u)$ et $B_{n,j}(v)$ sont des fonctions de degré m et de degré n, alors on dit que la surface de Bézier est de degré (m,n).

L'ensemble des points de contrôle est appelé réseau de Bézier ou réseau de contrôle. Les paramètres u et v sont dans l'intervalle [0,1].

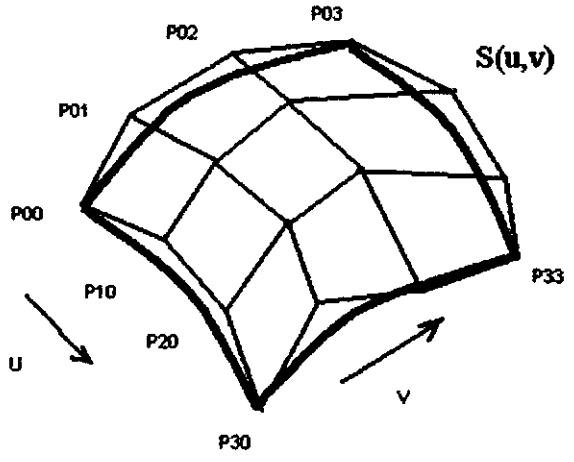


***Figure I.10** : Surface de Bezier et son réseau de contrôle*

- **Surface B-Spline :[3]**

Une surface B-Spline est définie à partir des informations suivantes :

-Un réseau de $(m+1)\times(n+1)$ points de contrôle $P_{i,j}$, où $0 \le i \le m$ et $0 \le j \le n$,

-Un vecteur des nœuds de h+1 nœuds dans la direction u, $U=\{0=u_0,u_1,\ldots,u_h=1\}$

-Un vecteur des nœuds de k+1 nœuds dans la direction v, $V=\{0=v_0,v_1,\ldots,v_k=1\}$

-Le degré p de la surface dans la direction u,

-Le degré q de la surface dans la direction v,

La *figure I.11* montre le réseau de contrôle d'une surface B-Spline

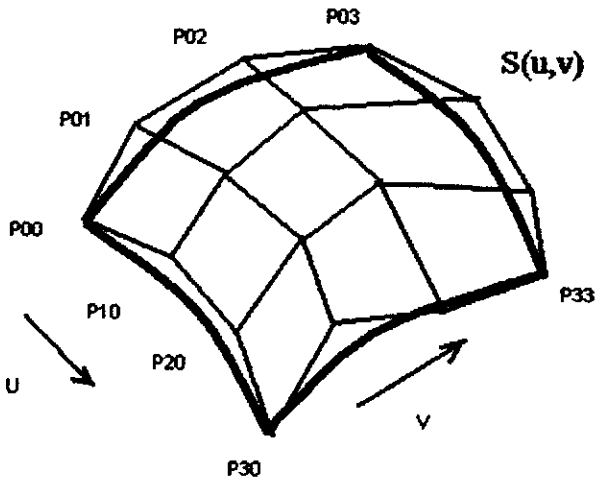


Figure I.11 : *Surface de B-Spline et son point de contrôle*

La surface B-Spline définie par ces informations est donnée par :

$$P(u,v)=\sum_{I=0}^{m}\sum_{j=0}^{n} N_{i,p}(u)\, N_{j,q}(v)\, P_{i,j}$$

Où : $N_{i,p}(u)$ et $N_{j,q}(v)$ sont les fonctions base B-Splines de degré p et q respectivement et sont données par :

$$N_{i,0}(u) = \begin{cases} 1 & si\ u_i \le u < u_{i+1} \\ 0 & sinon \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{U_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u)$$

$$N_{j,0}(u) = \begin{cases} 1 & si\ v_i \le v < v_{i+1} \\ 0 & sinon \end{cases}$$

$$N_{i,p}(u) = \frac{v - vj}{vj_{+p} - vj} N_{i,p-1}(v) + \frac{v_{j+q+1} - v}{v_{j+q+1} - v_{j+1}} N_{j+1,q-1}(v)$$

Les identités fondamentales, une pour chaque direction, doivent être satisfaites :

$$\begin{cases} h = m+p+1 \\ \text{et} \\ k = n+q+1 \end{cases}$$

Comme la surface de Bézier, la surface B-Spline est une surface tensorielle. L'ensemble des points de contrôle est le réseau de contrôle avec u et v sont dans l'intervalle [0,1].

- **Surface NURBS :[4]**

Une surface NURBS est définie à partir des informations suivantes :

- Un réseau de $(m+1)\times(n+1)$ points de contrôle $p_{i,j}$, où $0 \le i \le m$ et $0 \le j \le n$,
- Pour chaque point de contrôle est associé un poids $w_{i,j} \ge 0$,
- Un vecteur de $h+1$ nœuds dans la direction u, $U = \{0 = u_0, u_1, \ldots, u_h = 1\}$,
- Un vecteur de $k+1$ nœuds dans la direction v, $V = \{0 = v_0, v_1, \ldots, v_k = 1\}$,
- Le degré p de la surface dans la direction u,
- Le degré q de la surface dans la direction v.

La figure I.12 montre le réseau de contrôle d'une surface NURBS:

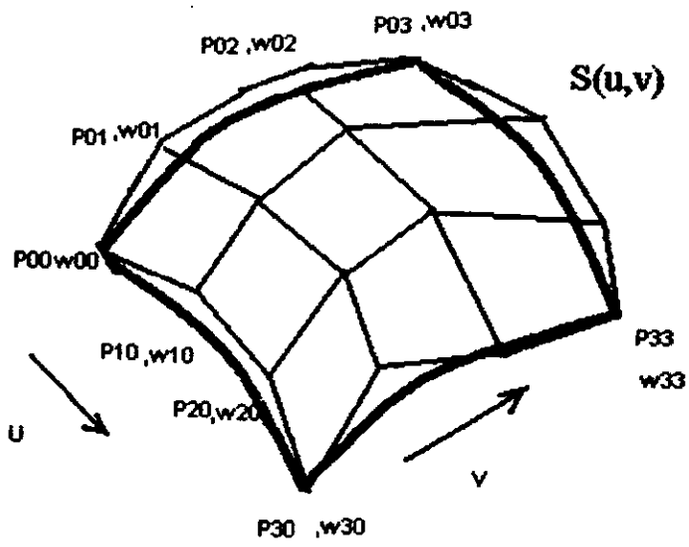


Figure I.12 : *Surface NURBS et son réseau de contrôle*

La surface NURBS définie par ces informations est donnée par :

$$S(u,v)=\frac{\sum_{i=0}^{m}\sum_{j=0}^{n}N_{i,p}(u)N_{j,q}(v)w_{i,j}p_{i,j}}{\sum_{i=0}^{m}\sum_{j=0}^{n}N_{i,p}(u)N_{j,q}(v)w_{i,j}}$$

où $N_{i,p}(u)$ et $N_{j,q}(v)$ sont les fonctions base B-Splines de degré *p* et *q* respectivement. Les identités fondamentales, une pour chaque direction, doivent être satisfaites :

$$\begin{cases} h = m + p + 1 \\ \quad \text{et} \\ k = n + q + 1 \end{cases}$$

Comme les surfaces de Bézier et B-Spline, une surface NURBS est une surface tensorielle. L'ensemble des points de contrôle est le réseau de contrôle, avec *u* et *v* sont dans l'intervalle [0, 1].

IV.2.2- Méthodes basées sur les courbes :

En plus des points, les courbes peuvent être aussi utilisées comme des contraintes lors de la conception des surfaces. La méthode qui utilise un ensemble de courbes données dans la phase de conception est appelée méthode de conception des surfaces par des coupes (Cross-Sectional Design).

Puisque le nombre de points de contrôle de plusieurs courbes est souvent plus petit que le nombre de points de contrôle d'une surface, la méthode de conception des surfaces par des coupes permet de remplacer la complexité de la conception d'une surface par la conception d'un ensemble de courbes en manipulant peu de points de contrôle.

Cette technique permet la génération des surfaces gauches, des surface de révolution, des surfaces oscillantes (swung surfaces), des surfaces balayées (swept surface) et des surfaces lissées (skinned surface).

- **Surface gauche :[5]**

Une surface gauche (surface réglée) est la surface la plus simple qui peut être construite par la méthode de conception par des coupes. La conception d'une surface gauche nécessite deux courbes $C_1(u)$ et $C_2(u)$, où u est un paramètre dans l'intervalle [0,1], et la surface est l'union des segments de droite $C_1(u)\ C_2(u)$. Plus précisément, par variation de u de 0 à 1, les segments C_1 (u) C_2 (u) génèrent une surface gauche.

La surface gauche s(u,v) définie par les deux courbes $C_1(u)$ et $C_2(u)$ est donnée par :

$$S(u,v) = v.\ C_1(u)+(1-v).\ C_2(u) \qquad u,v \in [0,1]$$

A partir de cette définition, on peut dire qu'une surface gauche est une interpolation linéaire entre deux courbes.

La figure I.13 montre les deux courbes utilisées et la surface générée.

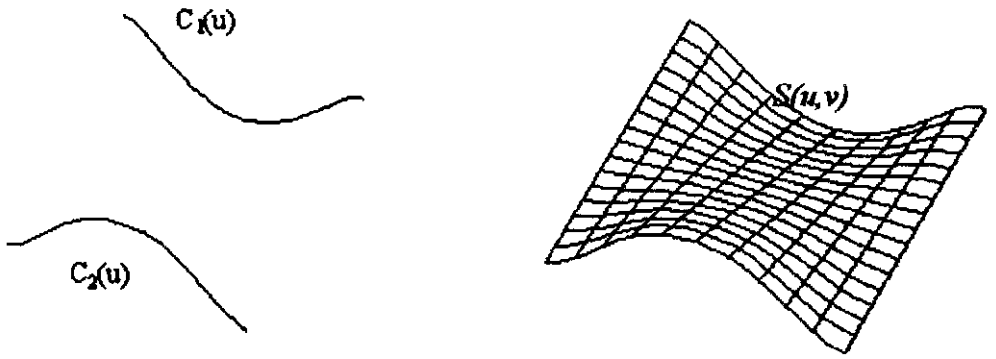


***Figure I.13** : Génération d'une surface réglée*

- **Surface de révolution :[1]**

Les surfaces de révolution sont des surfaces très utilisées en mécanique puisqu'elles peuvent être usinées facilement. La conception d'une surface de révolution nécessite une courbe C (u) représentant le profil de la pièce finie et une ligne représentant l'axe de rotation. La surface est obtenue en faisant tourner la courbe de profil autour de l'axe de rotation.

La surface de révolution générée S(u,v) est donnée par :

$$\mathbf{S(u,v)= M(v).C(u)} \qquad u,v \in [0,1]$$

Où *m(v)* est la matrice de transformation de rotation.

La figure I.14 montre la courbe de profil et la surface de révolution générée.

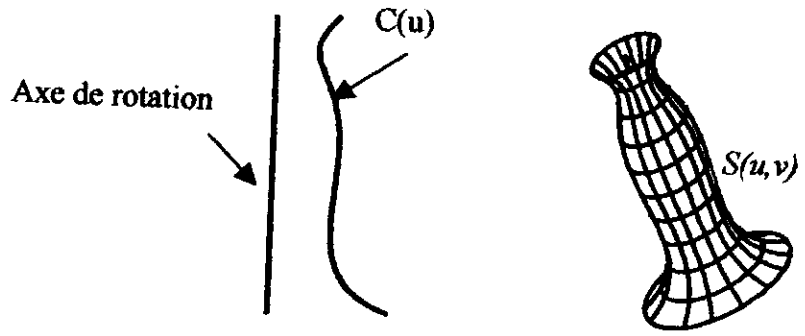


***Figure I.14** : Génération d'une surface de révolution*

- **Surface balayée :[1]**

La conception d'une surface balayée nécessite deus courbes, une courbe *C(u)* appelée courbe de profil et une courbe *T(v)* appelée courbe de trajectoire. La surface balayée est le résultat du balayage de la courbe *C(u)* le long de la courbe *T(v)*.

La surface balayée S(u,v) est donnée par :

$$\mathbf{S(u,v) = T(v) + M(v).\ C(u)} \qquad u,v \in [0,1]$$

Où *M(v)* est la matrice de transformation 3× 3 incorporant les rotations les changements d'échelle de *C(u)* comme une fonction de v. Pour chaque point sur la courbe de trajectoire *T(v)*, passe une courbe de profil *C(u)* transformée par la matrice *M(v)*.

La *figure I.15* montre la courbe de profil, la courbe de trajectoire et la surface balayée générée

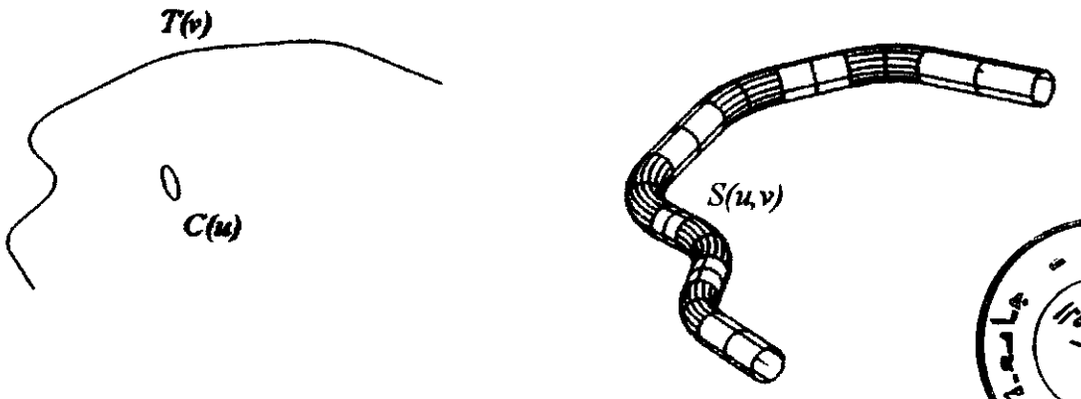


Figure I.15 : *Génération d'une surface balayée*

- **Surface oscillante (swung surface) :[4]**

Une surface oscillante est une généralisation des surfaces de révolution. La conception d'une surface oscillante nécessite deux courbes dans deux plans perpendiculaires, une courbe de profil *C(u)* dans le plan *xz* et une courbe de trajectoire *T(v)* dans le plan *xy* avec :

$$\begin{cases} C(u)=(C_x(u),0,C_z(u)) & u\in[0,1] \\ T(v)=(T_x(v),T_y(v),0) & v\in[0,1] \end{cases}$$

Dans ce cas, la surface oscillante S(u,v) est donnée par :

$$S(u,v)=(\alpha.\ C_x(u).T_x(v)\ ,\ \alpha.C_x(u).\ T_y(v)\ ,\ S_z(u)) \qquad u,v\in[0,1]$$

La figure I.16 montre la courbe de profil, la courbe de trajectoire et la surface oscillante générée

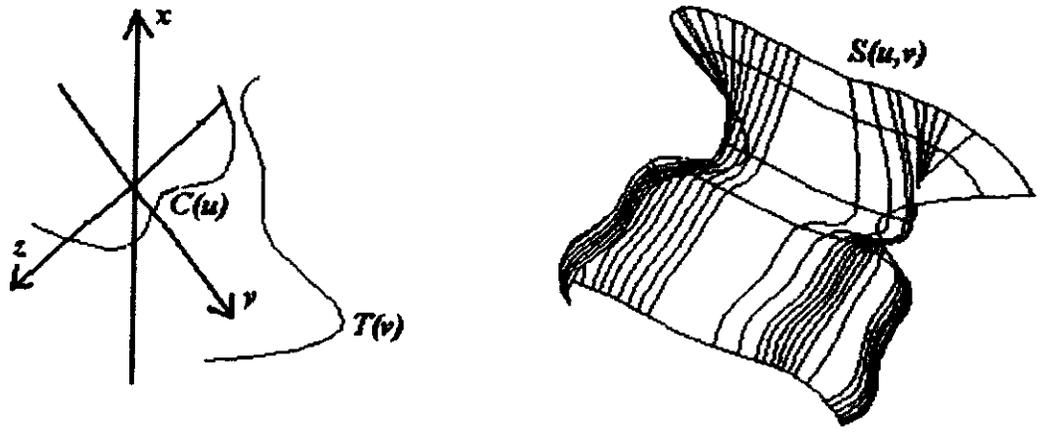


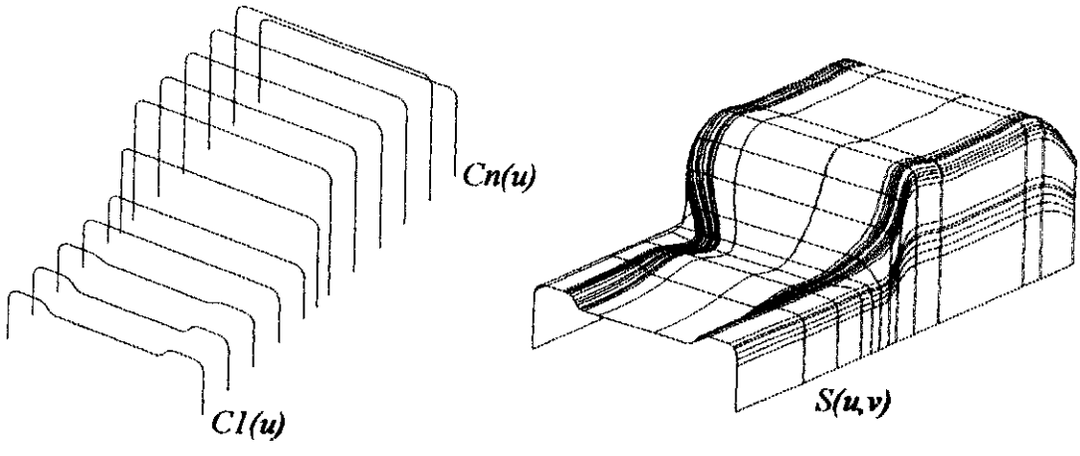
Figure I.16 : *Génération d'une surface oscillante*

- **Surface lissée (Skinned surface) :[5]**

Pour un ensemble de n courbes $C_1(u)$, $C_2(u)$,..., $C_n(u)$, il existe au moins une surface S(u,v) qui contient toutes les courbes . La surface S(u,v) est obtenue par une interpolation globale sur l'ensemble des courbes données.

La surface construite S(u,v) ne contient pas uniquement ces courbes , mais elles sont de plus des courbes isoparamétriques. Plus précisément, $S(u,v_i)= C_i(u)$ pour $0=v_1<v_2<\ldots<v_n=1$.

La figure I.17 montre les différentes courbes de profil et la surface lissée générée



Figurel.17 : *Génération d'une surface lissée*

- **Surface de Gordon :[4]**

La conception d'une surface de Gordon nécessite deux réseaux de courbes suivant deux directions. Le premier réseau de courbes est $F_k(u)$ *(0<= k<=k et u∈[0,1])*, tandis que le deuxième réseau est $G_i(v)$ (0<=I<=L et v ∈[0,1]). Chaque courbe F k (u) a un seul point d'intersection $Q_{i,k}$ avec $G_i(v)$. La surface construite *S(u,v)* interpole les

courbes F k(u) et $G_l(v)$, c'est-à-dire que les courbes $F_k(u)$ et $G_l(v)$ deviennent des courbes isoparamétriques de la surface.

La forme générale d'une surface de Gordon est la suivante :

$$S(u,v)= L1(u,v)+ L2(u,v)- T(u,v) \qquad u,v\in[0,1]$$

Où $L_1(u,v)$ est la surface qui interpole les courbes $F_k(u)$ dans la direction v , $L_2(u,v)$ est la surface qui interpole les courbes $G_l(v)$ dans la direction u et T(u,v) est la surface qui interpole les points d'intersection $Q_{l,k}$ des courbes données.

La figure I.18 montre le réseau de courbes utilisés et la surface de Gordon générée

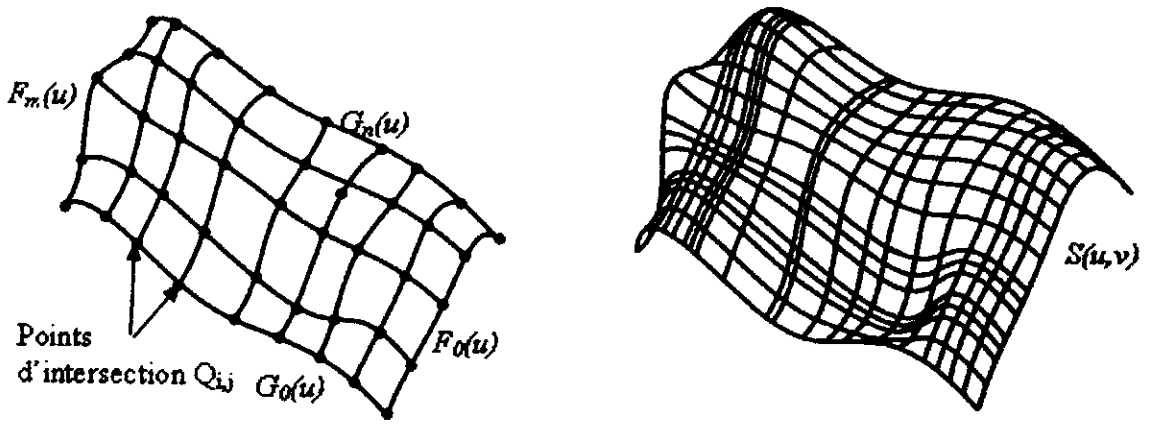


***Figure I.18** : Génération d'une surface de Gordon*

- **Surface de Coons :[1]**

La conception d'une surface de Coons nécessite quatre courbes qui représentent les courbes limites de la surface qu'on veut générer. La surface de Coons S(u,v) est un cas particulier de la surface de Gordon.

Les courbes frontières sont F0(u), F1(u), G0(v) et G1(v) avec u,v dans l'intervalle [0,1]. Ces courbes limites doivent satisfaire les égalités suivantes :

F0(0)= G0(0), F0(1)= G1(0), F1(0)=G0(1) et F1(1)= G1(1).

Une solution simple de ce problème est la surface bilinéaire S(u,v) de Coons et est donnée par :

$$S(u,v)=(1-u)\,S(0,v)+u\,S(1,v)+(1-v)\,S(u,0)+v\,S(u,1)$$

$$-[1-u \quad u]\begin{bmatrix} S(0,0) & S(0,1) \\ S(0,1) & S(1,1) \end{bmatrix}\begin{bmatrix} 1-v \\ v \end{bmatrix}$$

Où: $S(0,v) = G_0(v)$, $S(1,v)=G_1(v)$, $S(u,0)= F_0(u)$ et $S(u,1)= F_1(u)$.

La figure I.19 montre les courbes frontières et la surfaces de Coons générée

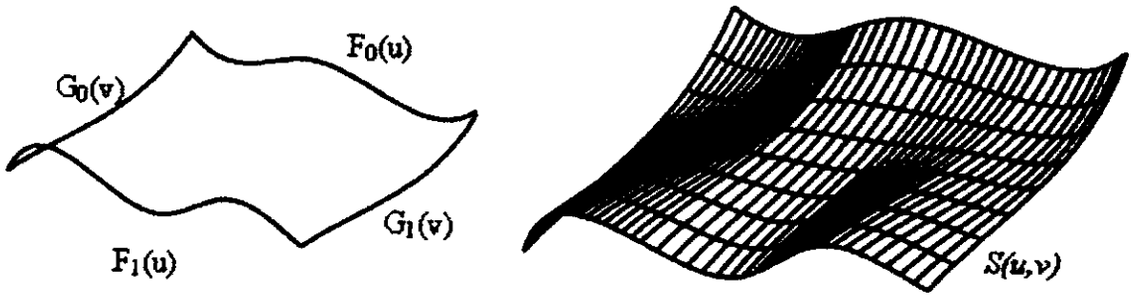


***Figure I.19** : Génération de Coons.*

- **Surface extrudée :[4]**

La surface extrudée peut être vue comme une extrusion d'une courbe suivant une direction donnée.

La figure I.20 illustre le principe de génération des surfaces extrudées.

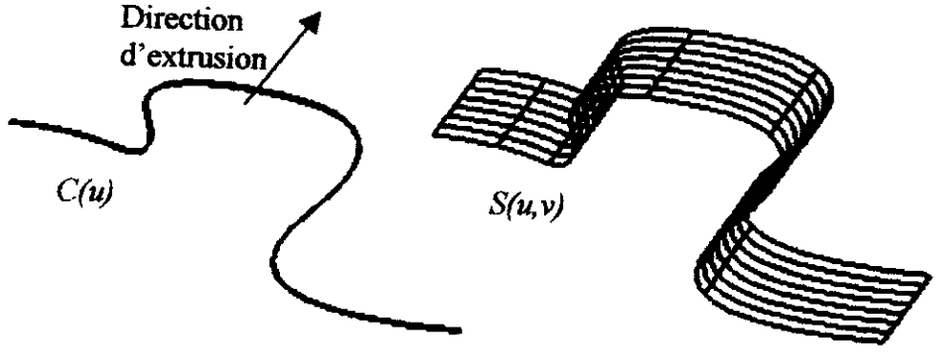


***Figure I.20** : Génération d'une surface extrudée*

V- CONCLUSION

Dans ce chapitre, nous avons mis l'accent sur les différentes formes de représentation des courbes irrégulières et les différentes méthodes de conception des surfaces complexes. Ces méthodes de conception sont divisées en deux classes :

- Conception basée sur un nuage de points. La surface la plus puissante est de type NURBS qui englobe les surfaces B-Spline et les surfaces de Bézier.
- Conception basée sur les courbes.

Le prochain chapitre sera consacré à l'étude des machines outils à commande numérique.

LES MACHINES OUTILS A COMMANDE NUMERIQUE

2

DANS CE CHAPITRE

- LES MACHINES OUTILS A COMMANDE NUMERIQUE (MOCN)
- LES FRAISEUSES NUMERIQUES

I- INTRODUCTION

Dans ce chapitre, nous allons décrire les machines outils à commande numériques (plus particulièrement les fraiseuses numériques) du point de vue modélisation géométrique et modélisation globale, puis nous nous intéressons à la classification de ces même machines selon deux points de vue :

- Une classification selon le nombre d'axes.
- Une classification selon la distribution des mouvements entre les différents organes mobiles de la machine.

II- LES MACHINES OUTILS A COMMANDE NUMERIQUE (MOCN)

II.1- Définition d'une MOCN :[6]

Une machine-outil à commande numérique (MOCN) est machine-outil programmable équipée d'une commande numérique par calculateur (CNC).

Elle représente le moyen de production le plus important des pièces mécaniques. Elle assure la réalisation automatisée des pièces en spécifiant les mouvements nécessaires dans un programme.

Elle a pour but de réaliser physiquement les mouvements de coupes nécessaires à l'obtention d'une surface par enlèvement de matière. Elle réalise le mouvement de coupe et le mouvement d'avance de l'outil par rapport à la pièce. De plus, elle permet l'obtention de pièces en respectant les spécifications fonctionnelles.

II.2- Structure physique d'une MOCN [6]

Si on ne s'intéresse qu'à la méthode d'enlèvement de matière par mouvement de rotation (fraisage), la machine doit avoir la structure suivante :

- Des systèmes, autant que nécessaire, assurant la mise en position de l'outil par rapport à la pièce et les mouvements d'avance. Ce sont ***les axes de la machine*** ;

- Un système qui réalise le mouvement de coupe par mise en rotation des outils ou de la pièce : ***la broche*** ;

- Un système de contrôle - ***commande***, qui permet le suivi automatique du programme de commande de la machine ;

- Un élément mécanique qui assure le lien entre ces systèmes : ***le bâti.***
A cela, il faut ajouter des éléments d'interfaces spécifiques à la production permettant la mise en position des outils et des pièces sur la machine.

- **Le bâti**

Le bâti assure le guidage des axes de mouvements, et l'agencement des autres organes de la machine. Pour assurer une géométrie correcte, et encaisser les actions mécaniques dues aux accélérations élevées des parties mobiles, le bâti doit être rigide et capable de limiter les déformations dues à la chaleur.

- **Les axes de déplacement**

Les axes de déplacement mettent en mouvement les parties mobiles des machines avec de fortes accélérations.

- **La broche**

La broche crée le mouvement de coupe nécessaire à l'usinage. Elle assure donc la mise en rotation de l'outil.
Cinématiquement, la broche est en liaison pivot avec le bâti ou le chariot. Dynamiquement, elle doit être très rigide, et stable thermiquement de façon à garantir la stabilité de la position relative de l'outil par rapport à la pièce durant l'usinage.

- **Le directeur de commande numérique :**

La commande numérique assure l'asservissement en position et en vitesse des déplacements des mobiles. C'est purement de la commande d'axe, avec un traitement numérique pour élaborer les consignes de commande en temps réel en fonction des paramètres de la trajectoire et de l'état de la chaîne d'action. Le directeur de commande assure entre autres les fonctions suivantes :

- interprétation du programme d'application,
- détermination des phases de travail (blocs exécutables),
- calcul des consignes successives sur la trajectoire,
- élaboration de l'écart de poursuite et des corrections nécessaires,
- gestion des données et des mesures,
- surveillance des erreurs.

De plus, il gère l'ensemble des fonctions séquentielles associées à la machine, soit directement, soit à travers un automate programmable.

- **Les porte –outils**

Les porte-outils ont pour fonction d'assurer la liaison entre l'outil et la machine. Suivant le mode d'usinage, ils supportent des sollicitations dynamiques différentes. Dans le cadre du fraisage, les porte-outils assurent la liaison au moyen d'un cône normalisé.

II.3- FONCTIONNEMENT D'UNE MOCN :

Une machine outil à commande numérique reçoit les instructions nécessaires à l'exécution des opérations (coordonnées des points définissant la trajectoire à décrire, vitesses d'avance et de coupe, nature du cycle demandé, numéro d'outil à utiliser, etc....) à partir d'un programme d'usinage.

Le système doit connaître les coordonnées instantanées des éléments mobiles suivant chacun des axes qu'il commande .

De la différence entre les coordonnées à atteindre et les coordonnées instantanées, résulte les ordres de mouvement, de ralentissement et d'arrêt, qui, transmis aux moteurs électriques ou hydrauliques équipant chacun des axes, provoquent le déplacement désiré.

II.4- LES AXES D'UNE MOCN [7]

Un système de coordonnées permet de repérer les positions et les déplacements d'un objet par rapport à un point origine.

- **Définition:**

Un système de coordonnées cartésiennes rectangulaires de sens direct est un trièdre constitué de trois axes linéaires *X*, *Y* et *Z* auxquels sont associés trois axes rotatifs *A*, *B* et *C*.

- **Orientation des axes d'une machine-outil:**

Les axes de la machine sont définis (par convention) de la manière suivante:

- L'axe Z est parallèle à la broche principale de la machine;
- L'axe *X* correspond au plus grand déplacement horizontal;
- L'axe *Y* forme le trièdre direct avec les axes X et Z.

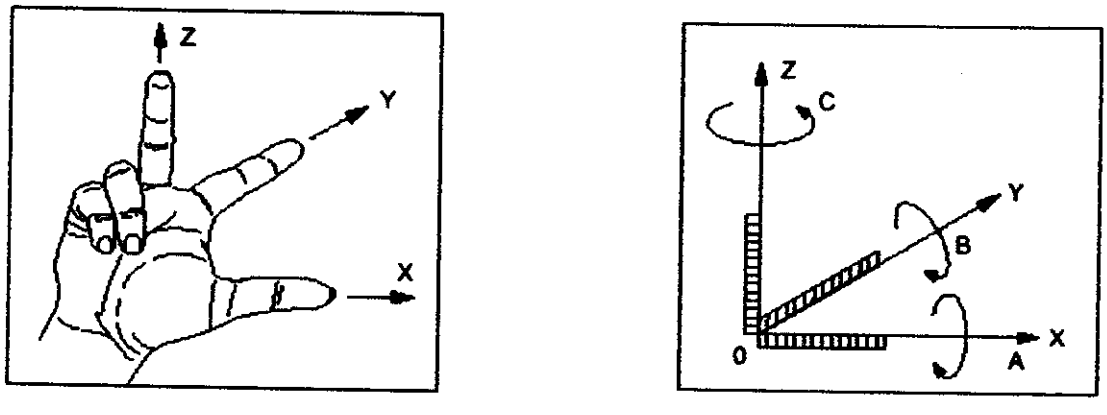


***Figure II.1** : Orientation des axes*

Les angles A, B et C définissent les mouvements de rotation effectués respectivement autour des axes X, Y et Z (figureII.1). Le sens positif des axes est défini de manière qu'un mouvement dans une direction positive d'axes de translation ou de rotation, augmente les valeurs positives de la position de la pièce par rapport à la machine.

- **Définition d'un axe numérique:**

C'est un axe de déplacement pour lequel une infinité de positions peut être atteinte à la résolution de positionnement près ou axe de déplacement asservi en position **et** en vitesse.

- **Définition d'un demi axe:**

Axe de déplacement pour lequel un ensemble fini de positions peut être atteint ou axe de déplacement asservi en position **ou** en vitesse.

- **Degrés de mobilité d'une machine-outil (nombre d'axes):**

Un degré de mobilité est un axe de mouvement. Le choix du nombre d'axes est dicté par la forme des pièces à usiner et de la cadence de production.
L'augmentation du nombre d'axes diminue le nombre de prises de pièce et de démontages nécessaires, et vice-versa.

II.5- MODELISATION GEOMETRIQUE D'UNE MOCN [6]

II.5.1- Pourquoi modéliser les MOCN ?

Par définition, le pilotage d'une machine-outil à commande numérique est réalisé par un ordinateur qui établit une stratégie de commande des axes (sortie) en fonction des déplacements mesurés de la machine (entrée). Pour pouvoir agir, le calculateur doit se représenter la machine dans l'espace en fonction des données (déplacements sur les axes). Il s'appuie sur le modèle. La modélisation mécanique de la partie opérative est nécessaire au calcul de l'asservissement. De même la modélisation géométrique de la machine est nécessaire à sa commande par l'opérateur.

II.5.2- Modélisation géométrique des MOCN : Cellule Elémentaire d'Usinage

Ce concept se veut un modèle général de description de la cinématique et de la géométrie d'une machine-outil. Directement dérivé de la robotique, il lui empreinte les dénominations et les méthodes de résolution.

- ***Principe de la cellule élémentaire d'usinage : CEU***

On considère que la situation relative de l'outil par rapport à la pièce est assurée par un «assemblage» de corps solides. La liaison entre ceux-ci peut être complète, mais démontable, ou autoriser un mouvement de rotation ou de translation, que ce soit un mouvement de réglage ou d'avance.

On distingue deux espaces de description : l'espace des tâches et l'espace articulaire. La géométrie des surfaces à usiner et les dimensions du porte-pièce sont décrites dans l'espace des tâches. C'est un espace à trois dimensions. Les mouvements relatifs des organes de la machine et leurs dimensions intrinsèques sont représentés dans l'espace articulaire. C'est un espace à autant de dimensions que d'axes de mouvement.

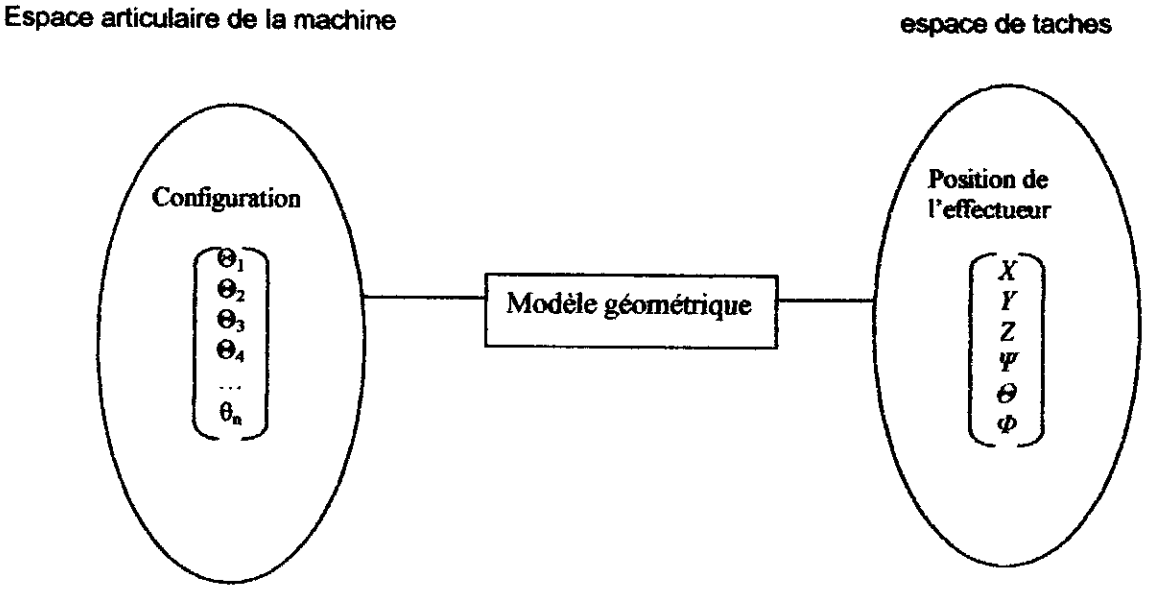


***Figure II.2** : Espace articulaire et espace des tache : représentation schématique*

- ***Mise en oeuvre du modèle***

Une fois les repères géométriques définis, on peut étudier les mouvements d'un point de l'espace des tâches dans l'espace articulaire. L'objet de la modélisation est de déterminer le modèle géométrique qui permet de passer d'une configuration de l'espace articulaire vers une position (de l'outil par rapport à la pièce) de l'espace des tâches.

Pour cela, on crée des modèles géométriques associés à chaque élément. Pour la chaîne cinématique, on utilise les modèles classiques de la robotique pour décrire les positions respectives des solides. Pour l'outil on définit un modèle caractérisant la partie active de l'outil. Enfin, on modélise les liaisons complètes par des matrices de changement de repère, ou des torseurs de petits déplacements.

Deux niveaux de finesse sont possibles pour modéliser la machine-outil.

- ***Modélisation globale de la machine***

La modélisation complète de la machine a pour but d'étudier précisément la phase d'usinage et les défauts géométriques associés. On ne prend en compte ni le type, ni la qualité, ni la position réelle des axes numériques de la machine. La machine est alors modélisée comme un changement de repère qui permet de passer d'un repère associé à l'outil à un repère associé à la pièce. On utilise pour cela les coordonnées homogènes, et les matrices de changement de repère.
L'expression au moyen de coordonnées homogènes offre la possibilité d'appliquer les mêmes opérateurs aux points et aux vecteurs. Ceux-ci sont différenciés par la valeur de la quatrième ligne, qui est un facteur d'échelle, égale à 1 pour les points, et

à 0 pour les vecteurs. Les opérateurs permettent d'exprimer une rotation, une translation ou les deux combinées.

$$M = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad \overrightarrow{OM} = \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix} \quad T(M) = \begin{bmatrix} 0\ 0\ 0\ tx \\ 0\ 0\ 0\ ty \\ 0\ 0\ 0\ tz \\ 0\ 0\ 0\ 1 \end{bmatrix} \quad R(M) = \begin{bmatrix} ax\ ay\ az\ 0 \\ bx\ by\ bz\ 0 \\ cx\ cy\ cz\ 0 \\ 0\ 0\ 0\ 1 \end{bmatrix} \qquad (1)$$

En coordonnées homogènes, le changement de repère s'applique de la manière suivante :

Soit R1 et R2 deux repères. On connaît la position du centre O2 du repère R2 dans le repère R1, et la matrice de changement de base de B1 associée à R1 à B2 associée à R2, c'est-à-dire les coordonnées des vecteurs de la base B2 exprimée dans B1.

$$\overrightarrow{O_1O_2} = \begin{bmatrix} tx \\ ty \\ tz \end{bmatrix}_{R_1} \qquad (B_2 \rightarrow B_1) = \begin{bmatrix} \alpha_i\ \alpha_j\ \alpha_k \\ \beta_i\ \beta_j\ \beta_k \\ \gamma_i\ \gamma_j\ \gamma_k \end{bmatrix}_{R_1} \qquad (2)$$

On a alors la relation suivante :

$$\overrightarrow{O_1M}_{R_1} = (R_1 \rightarrow R_2)\,\overrightarrow{O_2M}_{R_2} \qquad (3)$$

L'opérateur est :

$$(R_2 \rightarrow R_1) = \begin{bmatrix} \alpha_i\ \alpha_j\ \alpha_k\ tx \\ \beta_i\ \beta_j\ \beta_k\ ty \\ \gamma_i\ \gamma_j\ \gamma_k\ tz \\ 0\ 0\ 0\ 1 \end{bmatrix}_{R_1} \qquad (4)$$

- ***Modélisation par axe de la machine***

Dans ce cas, on cherche à obtenir un modèle qui permet le passage du repère associé à un solide, au repère associé au solide suivant. Ce passage est paramétré par deux distances et deux angles, qui évaluent les distances d_i, ai et les orientations $alpha_i$ et $teta_i$ entre les deux solides.

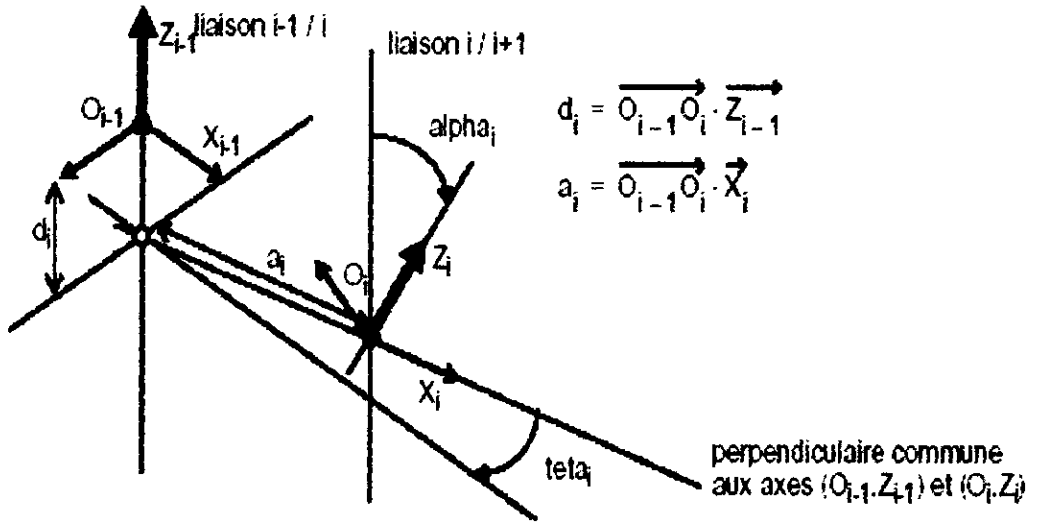


***Figure II.3** : Passage d'un solide d'une chaîne cinématique au suivant*

Ces deux niveau de finesse peuvent être utilisés en modélisation directe ou inverse :

- en ***modélisation directe***, pour un ensemble de consignes données aux axes, on trouve la position de l'extrémité outil dans le repère associé à la pièce ;

- en ***modélisation inverse***, pour un jeu de coordonnées de l'extrémité de l'outil exprimées dans le repère associé à la pièce, on trouve les consignes à programmer sur les axes. Ainsi, ce modèle est capable de produire par fermeture de la boucle un modèle de la surface usinée. On peut, donc, prévoir la qualité de la pièce avant usinage. Ce modèle permet aussi la transformation de coordonnées de l'espace des tâches à l'espace articulaire.

Les machines outils à commande numérique englobent entre autres les tours, les presses, les machines d'électroérosion et **les fraiseuses à commande numérique**. Ce sont ces dernières qui nous intéressent dans notre présente étude.

III- LA FRAISEUSE NUMERIQUE

III.1- DEFINITION :[8]

Une fraiseuse numérique est une machine outil à commande numérique à enlèvement des copeaux munie d'une broche tournante pouvant recevoir une fraise; outil à arêtes multiples ; ainsi que d'autre outils tels que foret, barre d'alésage, taraud permettent ainsi l' exécution d'usinage très variés.

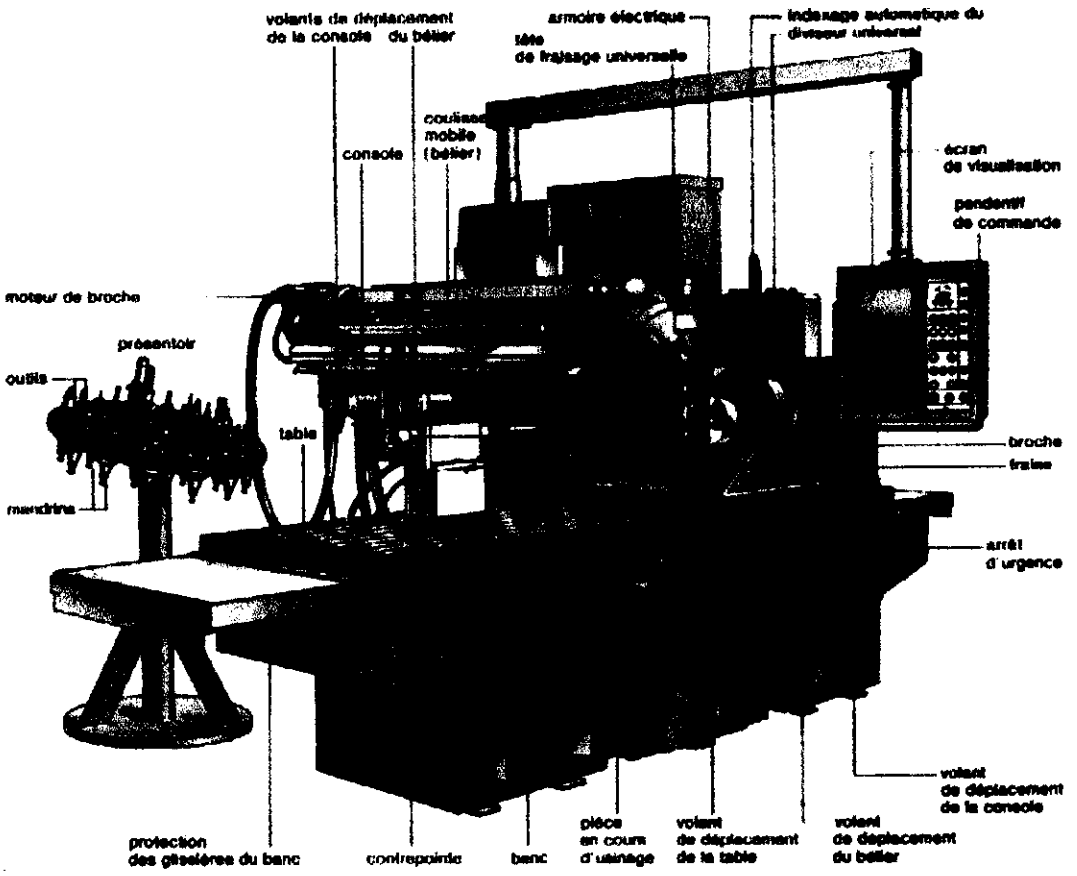


***Figure II.4** :Fraiseuse à commande numérique*

III.2- LES CONSTITUANTS D'UNE FRAISEUSE A COMMANDE NUMERIQUE

La fraiseuse à commande numérique est un ensemble qui comporte : **:[8]**

- ***La machine outil :***

La machine outil englobe la structure mécanique de la machine. Les différents mouvements sont commandés par des moteurs, et les déplacements sont contrôlés par des capteurs de mesure.

- ***Le directeur de commande numérique :***

Le directeur de commande numérique (DCN) est un automatisme composé d'éléments électroniques. Il a pour mission principale d'interpréter et de faire exécuter le programme pièce écrit dans un format normalisé, appelé **(format de programmation)**.

- ***L'armoire électronique :***

L'armoire électronique sert de relais entre la machine et le (DCN). Elle renferme des câbles, des amplificateurs et des fusibles.

- ***Le pupitre de commande :***

Le pupitre de commande sert à dialoguer avec le (DCN) et envoi des ordres de commande codés, il possède des touches sensitives, ainsi qu'un écran graphique qui sert à visualiser par exemple le programme, ou le profil fini de la pièce et la trajectoire des outils.

III.3- TYPES DE FRAISEUSES [9]

Plusieurs critères sont nécessaires pour caractériser une fraiseuse et ceux les plus souvent utilisés par les constructeurs sont indiqués ci-dessous :

- ***Par type d'usinage ou d'emploi*** :
  - fraiseuse ou fraiseuse-aléseuse.
  - Aléseuse-fraiseuse genre aléseuse possède une broche d'alésage coulissante.
  - Fraiseuse universelle et fraiseuse d'outillage possède le plus souvent un moyen d'orientation relative de la pièce et de la broche, ces machines sont en générale très maniables pour tout travaux unitaires variés.
  - Faiseuses spécialisées à prédominance de fraisage et vocation particulière (fraisage des rainures, des arbres à came....).
  - Centre d'usinage toujours caractérisé par la CN et le changement automatique de l'outil et éventuellement un dispositif de palettes.

- ***par architecture***
  - à console,
  - à banc,
  - à table croisé,
  - à table inclinable,
  - à table universelle (inclinable dans trois plan),
  - à montant mobile,
  - à montant fixe,
  - à portique mobile,

- à portique fixe,

- ***par position de la broche***
  - à broche horizontale
  - à broche verticale
  - à branche orientable (universelle, multiaxes)

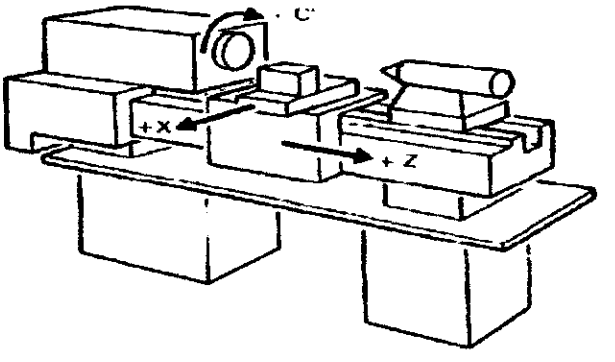


***Figure II.5** : Fraiseuse horizontale*

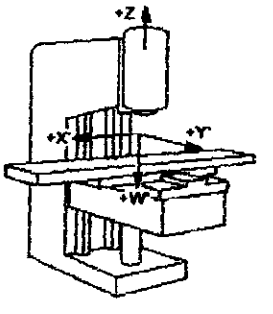


***Figure II.6** : Fraiseuse verticale*

- ***par dimensions***

  - par la course de déplacement sur l'axe X,
  - par la course de déplacement sur les trois axes principaux X, Y, Z,
  - pour les centre d'usinage à branche horizontale et palette, par les dimensions de la surface de la palette exprimées en millimètre.

III.4- La classification des fraiseuses numériques :

Traditionnellement, on a classé les machines en fonction des formes de surfaces à réaliser : (cylindriques / parallélépipédiques) (tournage / fraisage). Cette classification est remise en cause, car la commande numérique et l'adaptation des structures de machine cassent le lien entre les deux couples.
On classe maintenant les machines-outils par le nombre de mouvements élémentaires qu'elles peuvent mettre en oeuvre lors du déplacement de l'outil par rapport à la pièce. Seuls les axes sont décomptés. La mise en oeuvre simultanée de plusieurs outils entraîne l'augmentation du nombre d'axes. Cette classification ne permet pas d'associer directement un type de forme usinable à une classe de machine, car elle ne reflète pas la cinématique de l'outil. Par exemple un tour à cinq axes ne permet pas forcément de faire des pièces différentes par rapport à un tour à trois axes.**[6]**

On peut classer les fraiseuses en trois classes citées au-dessous :

- **Fraiseuses à 3 axes**

Les fraiseuses à trois axes se caractérisent par trois degré de liberté.
Le tableau suivant décrit les cas de fraiseuses de types trois translations.

Tableau II.1 : *les fraiseuses à trois axes (type trois translations)*

	broche	***Table***
1	0	X, Y, Z
2	X	Y, Z
3	Y	X, Z
4	Z	X, Y
5	X, Z	Y
6	Y, Z	X
7	X, Y	Z
8	X, Y, Z	0

Machines à 4 axes :

Ce type de fraiseuses se caractérise par quatre degré de liberté :

Le tableau suivant décrit les fraiseuses à 4 axes de type trois translations avec une rotation autour de l'axe **X**(A), et on peut avoir le même nombre de cas si on a une rotation autour de Y (B), ou bien autour de z (C).

***Tableau II.2** : les fraiseuses à 4 axes (trois translations et une rotation A)*

	broche	***table***
1	0	X,Y,Z,A
2	A	X, Y, Z
3	X	Y, Z, A
4	X ,A	Y, Z
5	Y	X, Z, A
6	Y ,A	X, Z
7	Z	X, Y, A
8	Z, A	X, Y
9	X, Z	Y, A
10	X,Z, A	Y
11	Y, Z	X, A
12	Y, Z, A	X
13	X, Y	Z, A
14	X, Y ,A	Z
15	X, Y, Z	A
16	X, Y, Z, A	0

Machines à 5 axes :

Ce type de fraiseuses se caractérise par cinq degré de liberté :

Le tableau suivant décrit les fraiseuses à 5 axes de type trois translations avec une rotation autour de X et l'autre autour de Y (**A et B**), et le nombre de cas sera le même pour les deux rotations (**B et C**) ou bien (**A et C**)

***Tableau II.3** : Les fraiseuses à cinq axes (trois translations et deux rotations A,B)*

	Broche	**Table**
1	0	X, Y, Z, A, B
2	X	Y, Z, A, B
3	Y	X,Z,A,B
4	Z	X, Y, A, B
5	X, Z	Y, A, B
6	Y,Z	X,A,B
7	X, Y	Z, A, B
8	X, A	Y,Z, B
9	X ,B	Y, Z, A
10	X, A, B	Y, Z
11	Y, A	X ,Z, B
12	Y ,B	X,Z, A
13	Y, A, B	X ,Z
14	Z, A	X,Y, B
15	Z,B	X,Y ,A
16	Z, A ,B	X,Y
17	X, Y, A	Z, B
18	X,Y, B	Z A
19	X,Y ,A,B	Z
20	X,Z, A	Y ,B
21	X,Z ,B	Y ,A
22	X,Z ,A,B	Y
23	Z,Y, A	X, B
24	Z,Y, B	X,A
25	Z,Y, A,B	X
26	X,Y, Z ,A	B
27	X,Y,Z, B	A
28	X,Y, Z, A,B	0
29	A	X,Y,Z, B
30	B	X,Y,Z, A
31	A,B	X ,Y, Z
32	X, Y, Z	A, B

III.5- Etude du couplet (usinage des surfaces gauches/types de fraiseuses) : [6]

Dans le tableau suivant , on présente les différentes configurations des fraiseuses numériques et on leur associe les types d'opérations d'usinage possibles.

Tableau II.4 :la classification des machines outils

Nombre d'axes	mouvements	Désignation du type d'usinage et d'opérations possibles
3	X,Y,Z	Fraisage : surfaçage, fraisage de poches, de rainures et de surfaces gauches. L'axe de l'outil reste parallèle à une direction fixe par rapport à la pièce.
4	X,Y,Z,B	Fraisage, surfaçage, perçage, fraisage de poches, de rainures et de surfaces gauches. L'axe outil reste contenu dans un plan fixe par rapport à la pièce.
4	X,Y,Z,C	Fraisage (cf (XYZB))
5	X,Y,Z,A,C	Fraisage de formes gauches, fraisage avec flanc de l'outil, fraisage avec d'épinçage, perçage en toute direction.
5	X,Y,Z,B,C	Fraisage de formes gauches (cf. (XYZAC))
5	X,Y,Z,A,B	Fraisage de formes gauches (XYZAC)

III.6- Avantages des fraiseuses à commande numérique :

Les fraiseuses à commande numérique ont des possibilités impressionnantes d'usinage par rapport aux fraiseuses conventionnelles grâce **[8]** :

- à la rapidité de lecture et traitement des informations ;
- à la rapidité de réponse des divers organes mobiles ;
- à sa conception et ses vitesses de déplacement ;
- à la précision que confère son architecture ;
- aux possibilités de réalisation d'un profil de pièce ;
- à la souplesse de changement de type de pièce et à la polyvalence des travaux permis;
- aux cycles programmés ;
- aux modifications des dimensions, de la coupe possible en cours d'usinage.

IV- CONCLUSION

Dans ce chapitre, nous avons présenté une vue générale de la machine outil à commande numérique, en décrivant ces différents constituants. , et en étudiant sa modélisation géométrique , et en particulier, nous avons mis l'accent sur les fraiseuses numériques en définissant ses différentes classes selon le nombre d'axes et selon la distribution des mouvements sur les organes mobiles de la fraiseuse.

Ce chapitre nous permettra d'introduire le chapitre suivant qui consiste à étudier l'usinage des surfaces gauches sur les fraiseuses numériques.

3

L'USINAGE DES SURFACES GAUCHES SUR LES MOCN

DANS CE CHAPITRE

- DEFINITION DE L'USINAGE
- PRINCIPE DE FRAISAGE
- LE FRAISAGE SUR DES FRAISEUSES A 3 AXES
- LE FRAISAGE SUR DES FRAISEUSES A 5 AXES
- ETUDE COMPARATIVE ENTRE L'USINAGE 3 AXES et 5 AXES
- ANALYSE D'USINABILITE D'UNE SURFACE
- LES INTERFERENCES
- L'USINABILITE DES SURFACES GAUCHES SUR LES FRAISEUSES A 3 AXES
- L'USINABILITE DES SURFACES GAUCHES SUR LES FRAISEUSES A 5 AXES

I- INTRODUCTION

Ce chapitre, sera consacré à l'étude de l'usinage des surfaces gauches sur les fraiseuses numériques, que ce soit l'usinage à trois axes ou bien à cinq axes.

Nous allons présenter les conditions d'usinabilité d'une surface sur une fraiseuse à commande numérique et citer quelques types d'interférences produites lors du fraisage.

II- DEFINITION DE L'USINAGE [7]

Le terme usinage désigne tous les moyens mis en œuvre pour obtenir une pièce ayant des cotes et des tolérances fixées à l'avance , à partir d'un élément brut , tel qu'un lingot, ou semi-fini, tel qu'une pièce venue de forge ou de fonderie.

III- PRINCIPE DE FRAISAGE [8]

Procédé d'usinage de formes généralement prismatiques utilisant des outils de coupe à dents multiples de forme circulaire appelés ' fraises'.

La fraise est animé d'un mouvement de rotation uniforme, elle est généralement fixe, en translation, dans l'espace .La pièce, rendue solidaire d'une table, est déplacée au moyen d'actionneurs en fonction de l'usinage à réaliser.

IV- LE FRAISAGE SUR DES FRAISEUSES A 3 AXES [9]

IV.1- Les fraises utilisées

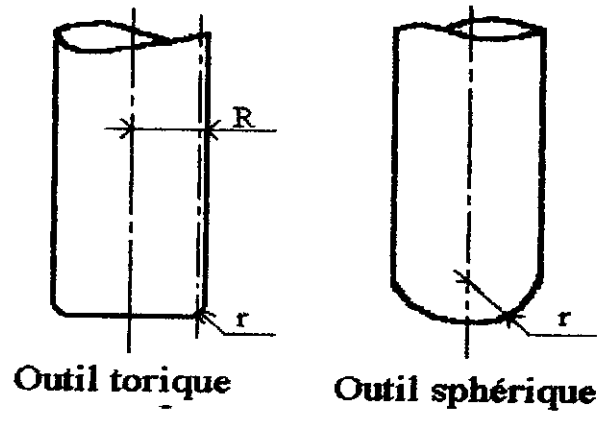


Figure III.1 : formes des outils

Si l'usinage des surfaces complexes par le procédé de copiage permettait d'éviter les collisions de l'outil avec la pièce, l'utilisation des Machines Outil à Commande Numérique MOCN nécessite un traitement en amont afin d'éviter tout risque de ce type.

L'usinage est réalisé à partir d'outils spécifiques tels que *la fraise torique* et *la fraise sphérique*. Le déplacement d'un point fixe situé sur l'axe de l'outil pendant l'usinage lorsque celui-ci se déplace sur toute la surface de la pièce est appelé ***surface centre outil***

La fraise sphérique apparaît comme un cas particulier de la fraise torique ou la côte R est nulle.

IV.2- Définition de la surface Centre Outil :

Nous appellerons la surface centre outil $S_0(U,V)$ la surface engendrée par un point fixe P_0 situé sur l'axe de l'outil , lorsque ce dernier se déplace sur toute la surface de la pièce .

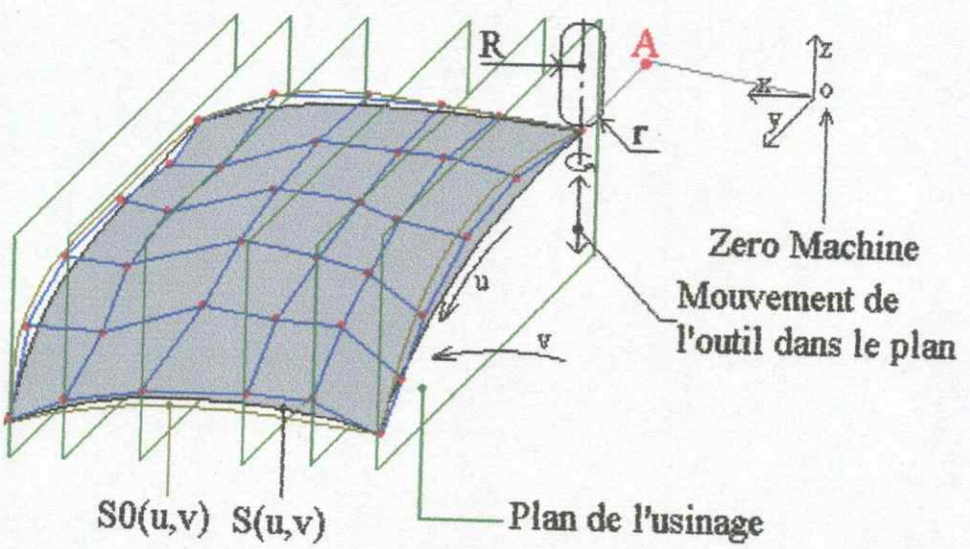


Figure III.2 : *La surface centre outil.*

Soit $S(u,v)$ l'équation de la surface de la pièce définie par les deux paramètres u et v variant respectivement dans les intervalles $[0,n]$ et $[0,m]$. L'usinage de la pièce est effectué à partir du zéro de la machine.

Les équations suivantes définissent la surface centre outil générée respectivement par un outil de coupe sphérique et torique :

S0(u,v)= S(u,v) +r* N(u,v). pour une fraise sphérique.

S0(u,v)= S(u,v) + r* Np(u,v) pour la fraise torique

Où :

N(u,v) : est le vecteur normale unitaire à la surface au point défini par u,v.
Np(u,v) : est le vecteur unitaire défini par la projection du vecteur N(u,v) dans le plan *Oxy*.
R,r : sont les caractéristiques de l'outil torique.

III.3- différents types d'usinages effectués sur les MOCN :

Il y a plusieurs façons d'usiner les formes complexes, les plus utilisées sont :

- ***Usinage par courbes isoparamétriques :***

Il consiste à utiliser l'équation de la surface centre outil S0(u,v) directement, pour cela on maintient un paramètre constant et on fait varier le deuxième paramètre sur son domaine de définition.

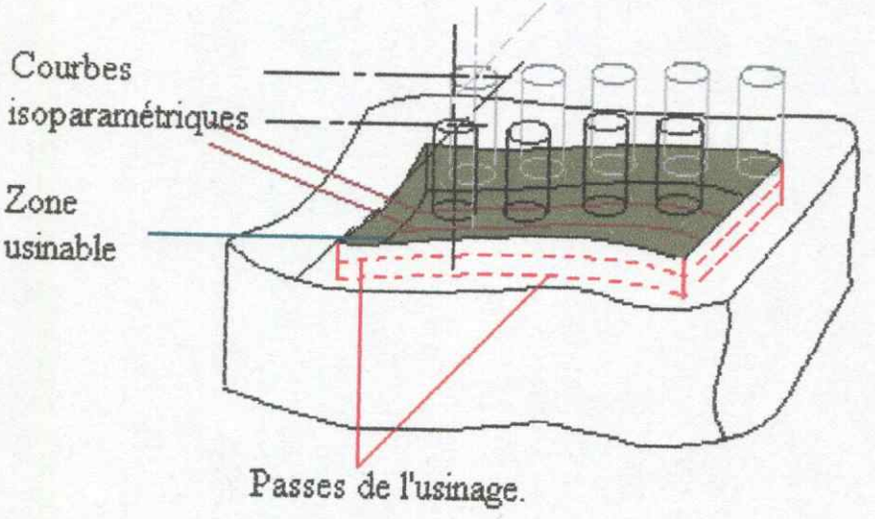


Figure III. 3 : *Usinage par courbes*

- ***Usinage par plans parallèles :***

Les surfaces gauches sont dans la quasi-totalité usinées par cette méthode qui consiste à déplacer le centre de l'outil dans des plans parallèles. L'usinage complet de la pièce nécessite un certain nombre de passages de fraises c-à-d un certain nombre de plans parallèles

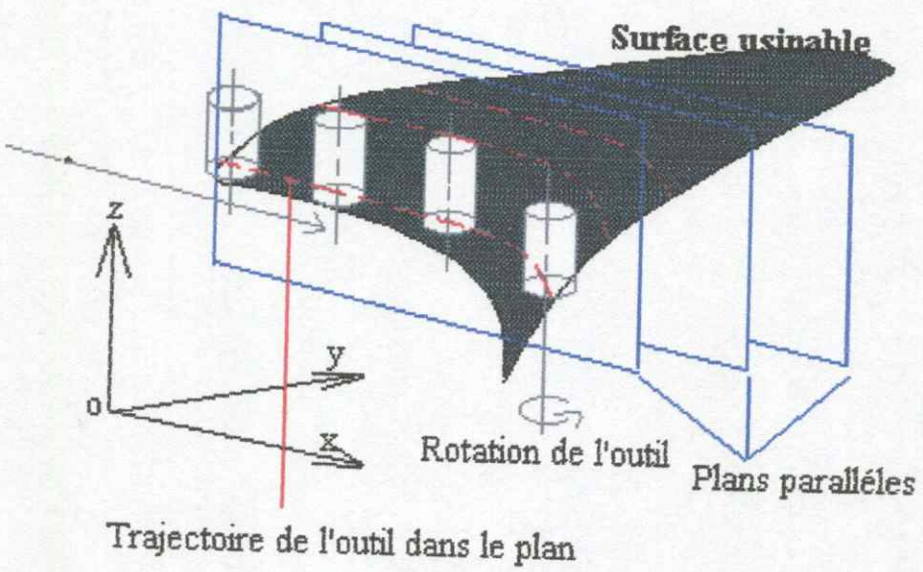


Figure III.4:*Usinage par plans*

- **Etude comparative :**

De point de vue de la modélisation, les surfaces de Bezier et de B-Spline sont les mieux adaptées aux surfaces peu irrégulières dans la forme, tel que les surfaces externes.

De point de vue de l'usinage, l'outil sphérique présente l'inconvénient d'avoir une vitesse de coupe nulle en son centre, ce qui supprime l'usinage en ce point. Par conséquent un mauvais état de surface est obtenu. La géométrie de la fraise torique remédie à ce problème dans la phase de finition de l'usinage.

L'usinage par courbes isoparamétriques est efficace dans les formes présentant des poches et des évidemment, par contre l'usinage par plans parallèles réalise un bon compromis entre les surfaces externes et les poches

Par définition, une fraiseuse à cinq axes offre deux degrés de liberté supplémentaires par rapport à une fraiseuse à trois axes. Ces deux degrés de liberté permettent d'orienter l'axe de l'outil.
Les machines à cinq axes sont utilisées pour produire les pièces de surfaces complexes comme : les hélices de turbine, les pièces aérospatiales et les moules ....Plus les pièces de surfaces gauches sont complexes plus on a besoin de ce type de fraiseuses.

V- LE FRAISAGE SUR DES FRAISEUSES A 5 AXES [10]

V.1- Les fraises utilisées pour l'usinage à 5 axes :

Trois types d'outils sont utilisés dans l'usinage à 5 axes : l'outil sphérique, l'outil torique, et l'outil cylindrique. Pour contrôler le fraisage on associe à l'outil un repère (X_t, Y_t, Z_t) . Ce repère se déplace sur la surface complexe à fraiser.
Nous allons présenter une description analytique des différents outils en terme de repère outil (X_t, Y_t, Z_t) .

- ***Fraise sphérique :*** On associe à cette fraise un repère (X_t, Y_t, Z_t) où :

X_t : est l'axe de la direction de l'usinage.
Y_t : correspond à l'axe de l'outil.
Z_t : c'est le troisième axe qui forme avec les deux autres un trièdre direct selon la règle de la main droite .
Les frontières de la fraise sphérique se caractérisent par deux formes essentielles : un quart de cercle et un segment droit *(figure III.5)*. La surface de cette fraise est décrite par la fonction suivante :

$$\Psi_{sphere}(\theta,\Phi,\beta)_T = \begin{pmatrix} R_2 \sin\theta \sin\Phi \\ R_2(1-\cos\theta)+\beta H \\ R_2 \cos\theta \sin\Phi \end{pmatrix}_T \qquad (1)$$

Avec : H: la longueur de la queue de l'outil.
R_2: le rayon de l'outil.
Θ :l'angle à partir de l'axe Z_T avec $\Theta \in [0,2\pi]$ (figure1).

- Quand $\beta=0$ et $\Phi \in [0,(\pi/2)]$, l'équation correspond au quart du cercle de la partie sphérique l'outil.

- Si $\Phi=\pi/2$ et $\beta \in [0,1.0]$, l'équation (1)correspond à la partie supérieure de l'outil sphérique(queue) . En utilisant l'équation (1) , le soin doit être pris de ne jamais permettre à β et Φ de changer simultanément dans leurs intervalles respectifs.

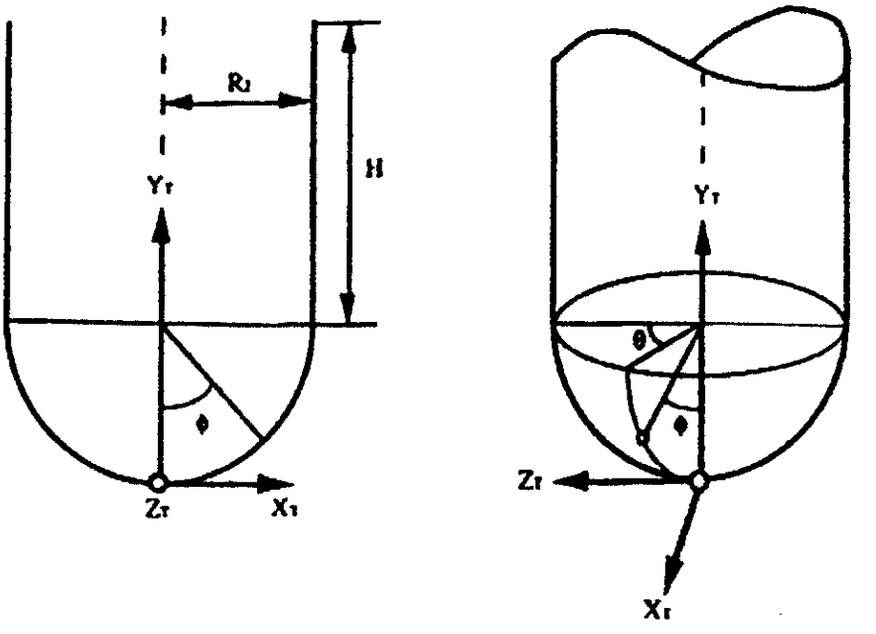


***Figure III.5** :Fraise sphérique dans un repère outil (X_t, Y_t, Z_t)*

- **Fraise cylindrique :**

La *figure III.6* montre la fraise cylindrique et le repère _outil correspondant. Cette fraise est décrite par :

$$\Psi_{Flat}(\theta,\alpha,\beta)_T = \begin{pmatrix} \alpha R_1 \sin\theta \\ \beta H \\ \alpha R_1 \cos\theta \end{pmatrix}_T \qquad (2)$$

Où H est la hauteur de la partie supérieure de l'outil, R_1 est le rayon de l'outil et θ est l'angle de l'axe Z_T avec $\Theta \in [0,2\pi]$.

- Quand β=0 et α ∈[0,1.0], l'équation ci-dessus correspond à la partie inférieure de l'outil cylindrique.
- Quand β ∈[0,1.0] et α=1, l'équation ci-dessus correspond à la partie supérieure de l'outil cylindrique,

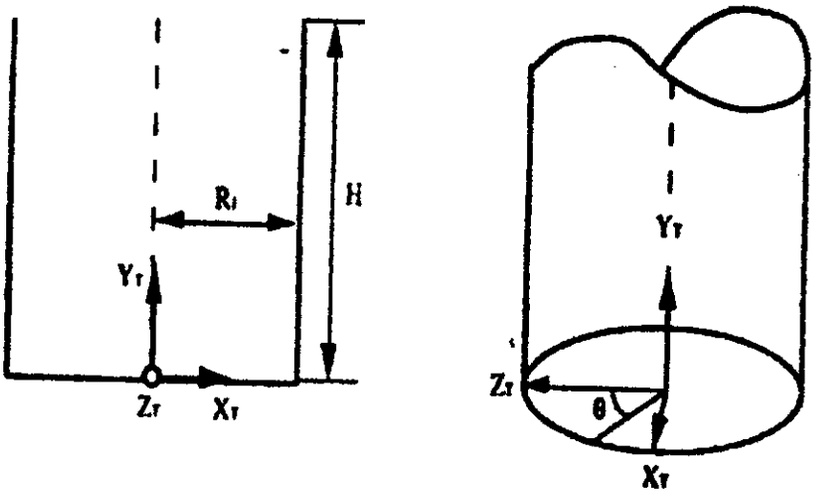


figure III.6 : Fraise cylindrique dans un repère outil (X_t, Y_t, Z_t)

- ***Fraise torique :*** L'outil torique est spécialement utilisé pour le fraisage à4 axes et à 5axes. Il se compose d'une surface plane qui sert de fond, un morceau d'un tore relié à un cylindre comme montré dans la *figure III.7* .La surface d'un outil torique peut être décrite comme suit :

$$\Psi_{Torus}(\theta,\Phi,\alpha,\beta)_T = \begin{pmatrix} (\alpha R_1 + R_2 \sin\Phi)\sin\theta \\ R_2(1-\cos\Phi) + \beta H \\ (\alpha R_1 + R_2 \sin\Phi)\cos\theta \end{pmatrix}_T \qquad (3)$$

Avec :
R_1 le rayon de la partie inférieure de l'outil.
R_2 est le rayon du tore,
Φ décrit le coin avec $\Phi \in [0, (\pi/2)]$,
θ est l'angle de l'axe Z_T avec $\Theta \in [0,2\pi]$ *(figure3).*

Dans l'équation (3), α décrit la partie inférieure de l'outil et β décrit la partie supérieure de l'outil.

L'équation 3 est une représentation générale pour les trois types d'outils. Les deux autres descriptions Ψ_{Flat} et Ψ_{sphere} en équation (1) et (2) sont des cas spéciaux de la description Ψ_{Torus} de l'équation (3).

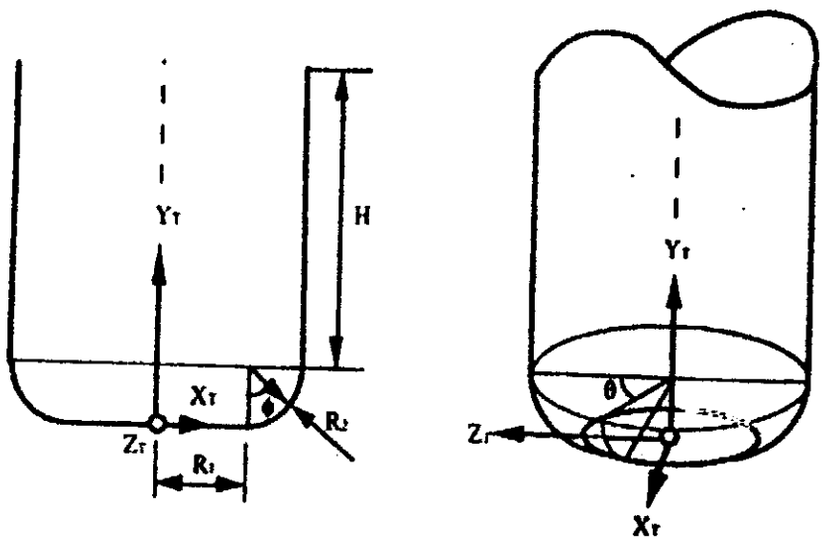


***Figure III.7** : Fraise torique dans un repère outil (X_t, Y_t, Z_t)*

- Si R_2 =0 et Φ=0, alors l'équation 3 est équivalente à Ψ_{Flat} correspondante à l'outil cylindrique décrite à l'équation (2).
- Si R_1=0 et α=0 l'équation (3) est équivalente à Ψ_{sphere} correspondante à l'outil sphérique décrite à l'équation (1).

Ainsi, $\Psi(\theta,\Phi,\alpha,\beta)^G_T$ est employée pour représenter la description générale des différents outils : Ψ_{Torus}, Ψ_{Flat}, Ψ_{sphere}

V.2- Description générale des outils inclinés dans l'usinage à 4 axes et à 5 axes

Dans l'usinage à 4axes, l'outil peut être orienté par une rotation autour d'un axe avec un angle d'inclinaison λ_L.

Comme montré dans la *figure III.8*, un repère local (X_L , Y_L , Z_L) est défini au point (cc), point de contact (outil / pièce): l'axe X_L est défini comme la direction d'usinage instantanée, Y_L est défini comme la normale de la surface et Z_L forme avec les deux autres axes un trièdre direct selon la règle de la main droite . Ce repère se déplace le long de la surface gauche à fraiser.

En usinage 4 et 5 axes, les positions de l'outil de coupe sont définies par des rotations autours des axes de déplacement en plus de l'inclinaison le long de l'axe Z_L,et l'inclinaison le long de l'axe Y_L avec un angle W_L.

Pour décrire un outil incliné, La formule générale de l'outil $\Psi(\theta,\Phi,\alpha,\beta)^G_T$ peut être trouvée en multipliant l'équation de la surface d'un outil torique « équation (3) » par la matrice de rotation (décrivant la rotation avec λ_L le long de l'axe Z_L),et la rotation

avec W_L le long de l'axe Y_L, et par la matrice de translation du repère outil (X_t, Y_t, Z_t) au repère locale(X_L, Y_L, Z_L) :

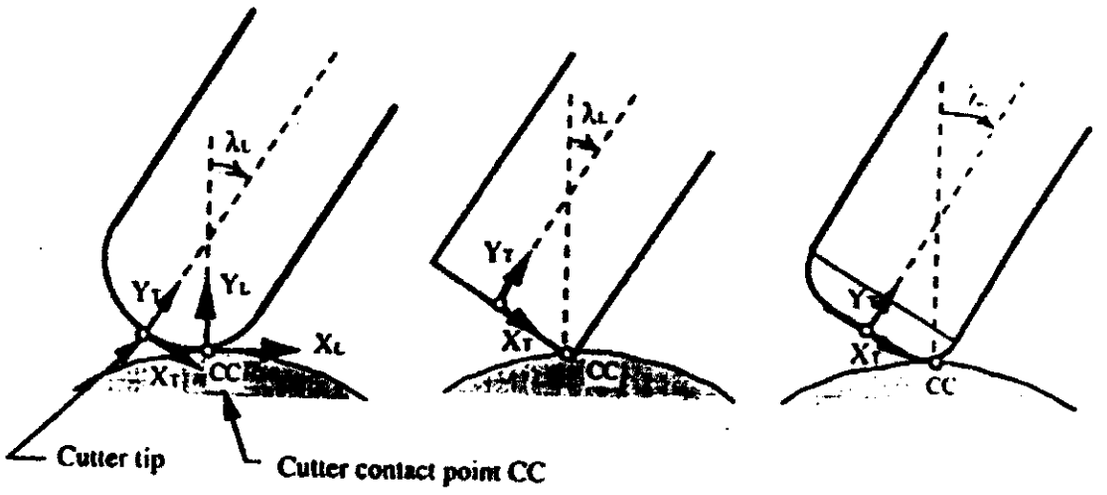


***Figure III.8** : Les trois types de fraises inclinées avec un angle d'inclinaison*

$$\Psi(\theta,\Phi,\alpha,\beta,\lambda_L,W_L)^G{}_T = Rot(-W_L).\ Rot(-\lambda_L).\ Tran \begin{pmatrix} -R_1-R_2\sin\lambda_L \\ -R_2(1-\cos\lambda_L) \\ 0 \end{pmatrix}$$

$$X\ \Psi(\theta,\Phi,\alpha,\beta)^G{}_T$$

$$= \begin{pmatrix} \cos W_L & 0 & -\sin W_L \\ 0 & 1 & 0 \\ \sin W_L & 0 & \cos W_L \end{pmatrix} \begin{pmatrix} \cos\lambda_L & \sin\lambda_L & 0 \\ -\sin\lambda_L & \cos\lambda_L & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} (\alpha R_1+R_2\sin\Phi)\sin\theta - R_1 - R_2\sin\lambda_L \\ R_2(1-\cos\Phi)+\beta H - R_2(1-\cos\lambda_L) \\ (\alpha R_1+R_2\sin\Phi)\cos\theta \end{pmatrix} \qquad (4)$$

Dans l'équation (4), la fonction $\Psi(\theta,\Phi, \alpha ,\beta, \lambda_L, W_L)^G{}_T$ est une description générale d'un outil orienté suivant deux angles (λ_L, W_L) dans un repère local pour un usinage à 5axes.

Pour l'usinage à 4 axes, la formule de l'outil $\Psi(\theta,\Phi, \alpha ,\beta, \lambda_L)^G{}_{T,\ 4axes}$ est un cas particulier de la description généralisée de l'outil $\Psi(\theta,\Phi, \alpha ,\beta, \lambda_L, W_L)^G{}_T$ de l'équation (4) en prenant $W_L=0°$

On suppose qu'on utilise l'outil torique, pour cela on prend $\alpha=1$ et $\beta=0$ pour décrire le tore de l'outil et on substitue ces deux paramètres dans la formule générale $\Psi(\theta,\Phi, \alpha ,\beta, \lambda_L, W_L)^G{}_T$ dans l'équation (4). Les points de surface (X_L ,Y_L , Z_L) de l'outil torique peuvent être représentés comme suit :

$$\Psi(\theta,\Phi, \lambda_L, W_L)^G{}_T= \begin{pmatrix} X_L \\ Y_L \\ Z_L \end{pmatrix}_{\Psi,L} \qquad \text{avec } \alpha=1 \text{ et } \beta=0 \qquad (5)$$

D'après l'équation (4), les coordonnées (X_L ,Y_L , Z_L) dans l'équation (5) du points du tore sont :

$X_L = m_1 \sin\theta + m_2 \cos\theta + m_3 \sin\Phi \sin\theta + m_4 \cos\Phi + m_5 \sin\Phi \cos\theta + m_6$ (6)

Ou :

$m_1 = R_1 \cos W_L \cos\lambda_L$
$m_2 = R_2 \sin W_L$
$m_3 = R_2 \cos W_L \cos\lambda_L$
$m_4 = - R_2 \cos W_L \sin\lambda_L$
$m_5 = - R_2 \sin W_L$
$m_6 = - R_1 \cos W_L \cos\lambda_L$

$Y_L = m_7 \sin\Phi \sin\theta + m_8 \sin\theta + m_9 \cos\Phi + m_{10}$ (7)

Ou

$m_7 = - R_2 \sin\lambda_L$
$m_8 = - R_1 \sin\lambda_L$
$m_9 = - R_2 \cos\lambda_L$
$m_{10} = R_1 \sin\lambda_L + R_2$

$Z_L = m_{11} \sin\Phi \sin\theta + m_{12} \sin\Phi \cos\theta + m_{13} \sin\theta + m_{14} \cos\theta + m_{15} \cos\Phi + m_{16}$ (8)

Avec :

$m_{11} = - R_2 \sin W_L \cos\lambda_L$
$m_{12} = R_2 \cos W_L$
$m_{13} = R_1 \sin W_L \cos\lambda_L$
$m_{14} = R_1 \cos W_L$
$m_{15} = - R_2 \sin W_L \sin\lambda_L$
$m_{16} = - R_1 \sin W_L \cos\lambda_L$

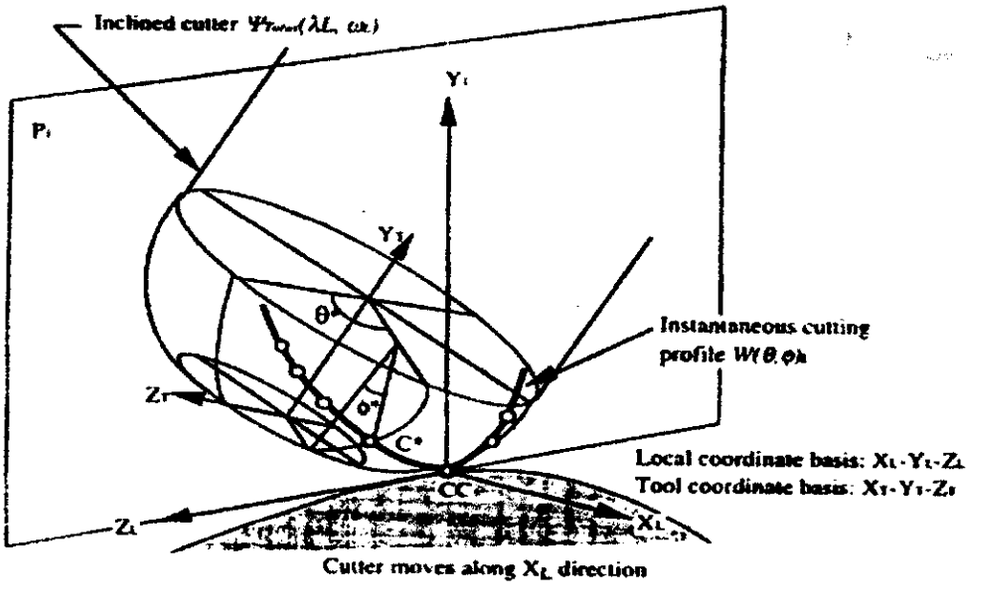
Tous les points de la surface (X_L ,Y_L , Z_L) de la formule générale $\Psi(\theta,\Phi, \lambda_L, W_L)^G{}_L$ d'un outil incliné avec une orientation (λ_L, W_L)peuvent être trouvés en utilisant les équations (6), (7) et (8).

V.3- Le profil de découpage instantané des outils inclinés pour un usinage à 5axes :

Pour éviter les interférences durant l'usinage multi-axes, il est important d'analyser les formes de coupes instantanées. Une coupe instantanée $W(\theta,\Phi)_L$ est définie par l'intersection entre un outil orienté $\Psi(\theta,\Phi, \lambda_L, W_L)^G_L$ et le plan Y_LZ_L . Il est difficile de l'évaluer dans le cas de l'usinage à 5 axes, à cause des deux mouvements simultanés de rotations supplémentaires.

Dans le plan Y_LZ_L, l'élément X_L de $\Psi(\theta,\Phi, \lambda_L, W_L)^G_L$ est égal à 0 ($X_L=0$). L'intersection entre le plan Y_LZ_L et l'outil $\Psi(\theta,\Phi, \lambda_L, W_L)^G_L$ qui est orienté avec (λ_L, W_L) peut être représentée comme suit :

$$W(\theta,\Phi)_L = \begin{pmatrix} X_L=0 \\ Y_L \\ Z_L \end{pmatrix} = \Psi(\theta,\Phi, \lambda_L, W_L)^G_{L,x=0} \quad \text{avec } \lambda_L, W_L \text{ données} \qquad (9)$$



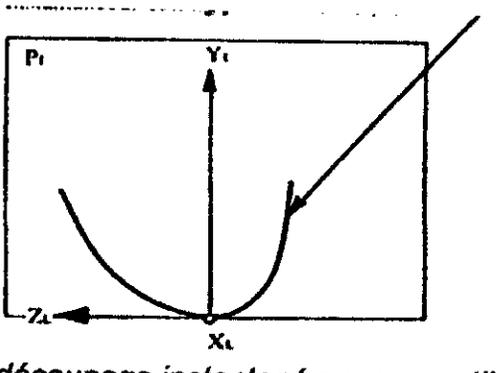


***figure III.9** : Le profile du découpage instantané pour un outil incliné en intersection avec le plan y_Lz_L*

Pour résoudre l'équation (9), les solutions qui satisfont la condition de $X_L=0$ sont en premier lieu recherchées. En utilisant l'équation (6) et $X_L=0$, pour un angle donné θ^*, on trouve une solution Φ^* pour le profil de découpage instantané $W(\theta,\Phi)_L$.

$$\Phi^* = \cos^{-1}\left(\frac{-n_2 n_3 + (n_1^4 + n_1^2 n_2^2 - n_1^2 n_3^2)^{1/2}}{n_1^2 + n_2^2} \right) \qquad (10)$$

avec:
$n_1 = m_3 \sin\theta^* - m_5 \cos\theta^*$
$n_2 = m_4$
$n_3 = m_1 \sin\theta^* + m_2 \cos\theta^* + m_6$

En remplaçant l'angle Φ par sa valeur Φ dans les équations (6), (7) et (8) le profile instantané $W(\theta^*,\Phi^*)_L$ de découpage peut être trouvé comme suit :

$$W(\theta^*,\Phi^*)_L = \begin{bmatrix} x^* \\ y^* \\ Z^* \end{bmatrix}^G_{\Psi,L}$$

$$= \begin{pmatrix} 0 \\ m_7 \sin\theta^* \sin\Phi^* + m_8 \sin\theta^* + m_9 \cos\Phi^* + m_{10} \\ m_{11} \sin\Phi^* \sin\theta^* + m_{12} \sin\Phi^* \cos\theta^* + m_{13} \sin\theta^* + m_{14} \cos\theta^* + m_{15} \cos\Phi^* + m_{16} \end{pmatrix} \qquad (11)$$

Où les constantes $m_7, m_8, \ldots\ldots, m_{16}$ sont définis dans les équations (7)à et (8).
En utilisant l'équation (11), le profil de découpage instantané peut être trouvé pour un outil torique orienté avec (λ_L, W_L) dans le cas de l'usinage à cinq axes.

V.4- Le rayon effectif d'un découpage avec un outil torique :

Dans l'usinage des surfaces gauches, il y a deux erreurs principales : creusage à la gouge et dégagement comme montré dans la figue 7.

Si l'outil pénètre la pièce avec une erreur négative pendant l'usinage et que cette erreur est plus grande que la tolérance extérieure exigée, on parle alors d'***un creusage à la gouge*** (gouging). Si des surépaisseurs (erreur positive) sont laissées sur la surface de pièce après l'usinage, on parle alors ***des dégagements*** ou ***les tranchants*** (undercut). Ces tranchants doivent être enlevés par d'autres procédés tels que le meulage ou le polissage.

La *figure III.15* illustre les deux types d'erreurs pendant l'usinage.

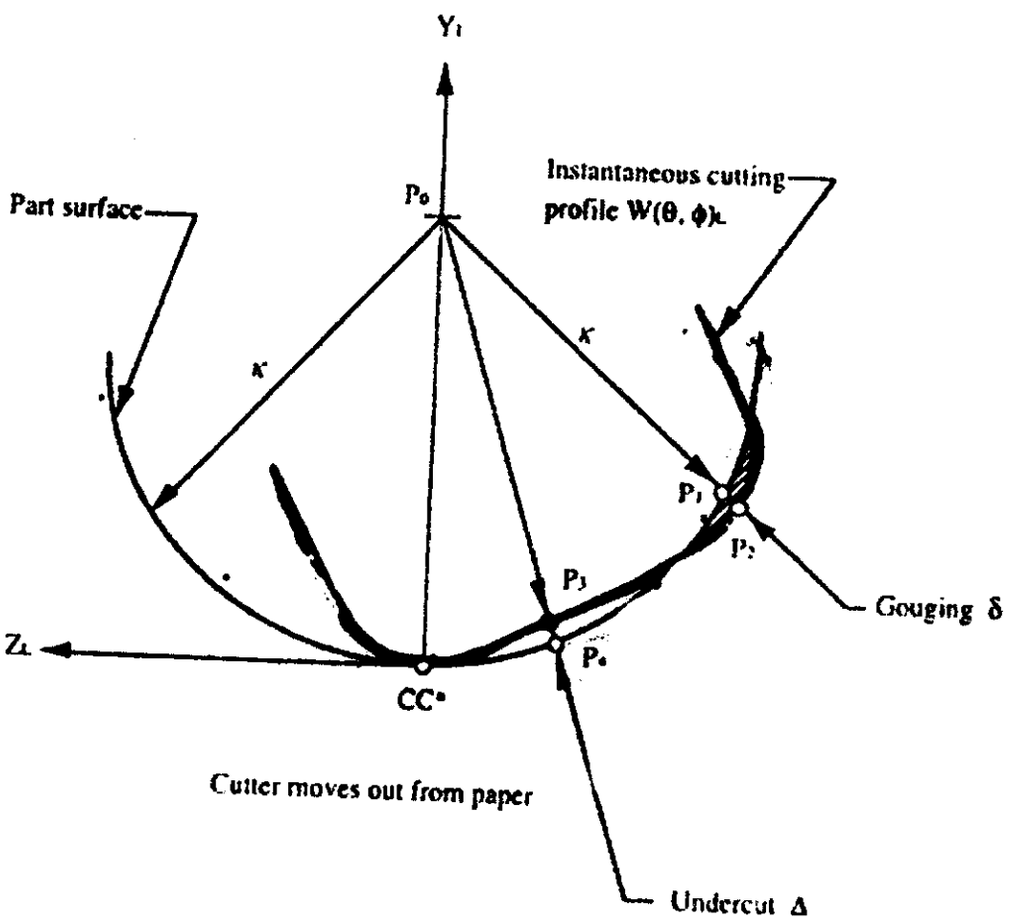
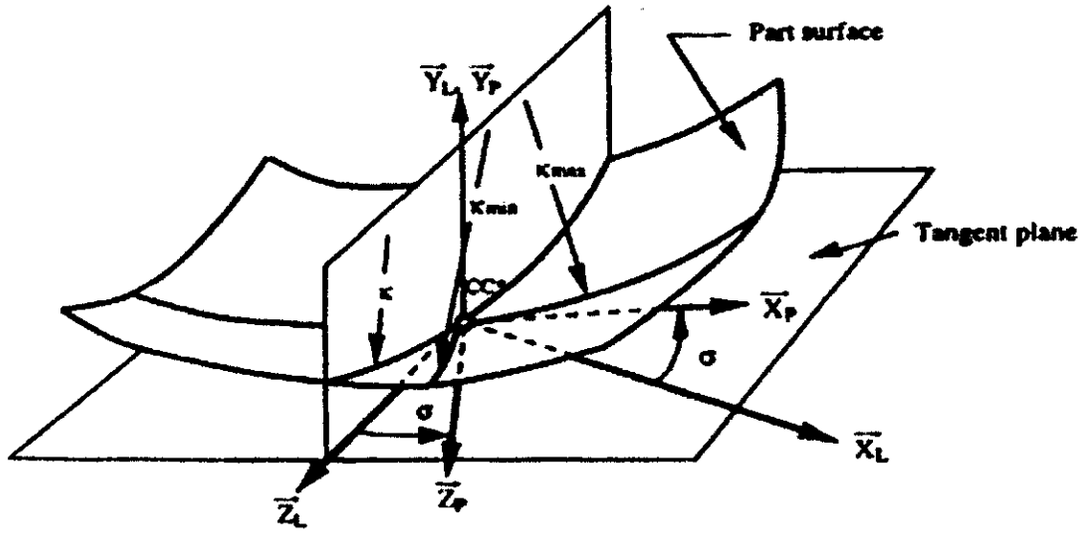


Figure III.10 *:Les erreurs des surfaces usinables*

Pour les surfaces gauches, lorsqu'on conçoit la trajectoire de l'outil, on prend toujours comme objectif d'éviter d'entrer en collision avec la surface et de réduire la taille des tranchants. Dans l'usinage à 5 axes, pour réduire la taille du tranchant laissé sur la surface usinée, le profil du découpage instantané $W(\theta^{\circ},\Phi^{\circ})_L$ devrait être aussi étroit que possible sur la surface de la pièce .



X_P-Y_P-Z_P: Principal coordinate system

X_L-Y_L-Z_L: Local coordinate system

Principal curvatures: ρ_{max}, ρ_{min}

$$k_{min} = \frac{1}{\rho_{min}}$$

figure III.11 : *La courbure de la surface locale sur le plan Y_L Z_L*

Pour éviter d'entrer en collision avec la surface à usiner , le rayon du découpage effectif $R_{\psi,eff}$ du profil du découpage $W(\theta^{\circ},\Phi^{\circ})_L$ ne doit pas être plus grand que le rayon K de la courbure locale (k=[1/ρ]. La figure (8) montre que la courbure locale ρ sur le plan Y_LZ_L peut être calculée comme suit :

$$\rho_{XL=0} = \rho_{min}\cos^2\sigma + \rho_{max}\sin^2\sigma \qquad \text{avec } \sigma = \cos^{-1}(\overrightarrow{X_p}\,.\,\overrightarrow{X_L}). \qquad (12)$$

En un point de contact (CC) localisé sur l'outil torique, les deux courbures principales $\rho_{\psi,min}$ et $\rho_{\psi,max}$ du tore de l'outil torique peuvent être trouvées en inclinant l'outil avec une angle λ_L non nul (et W_L=0) « figure (9) ». La courbure maximale peut être représentée par :

$$\rho_{\psi,max} = 1/R_2 \qquad \text{ou : } R_2 \text{ est le rayon du coin.} \qquad (13)$$

La courbure minimale est :

$$\rho_{\psi,min} = \rho(\theta = \pi/2, \lambda_L, W_L=0)_{CC} = \frac{\sin \lambda_L}{(R_1+R_2)\sin \lambda_L} \qquad (14)$$

Le rayon du découpage effectif $R_{\psi, Eff, XL=0}$ d'un outil torique dans le plan Y_L, Z_L ($X_{L=0}$) peut être trouvé comme suit :

$$R(\lambda_L, W_L)_{\psi, Eff, XL=0} = \frac{R_2(R2+ R_2\sin \lambda_L)}{R_2\sin \lambda_L \cos^2 W_L + (R_1+ R_2 \sin \lambda_L)\sin^2 W_L} \qquad (15)$$

Le rayon effectif du découpage $R_{\psi, Eff, ZL=0}$ d'un outil torique dans le plan X Y est décrit comme suit :

$$R(\lambda_L, W_L)_{\psi, Eff, ZL=0} = \frac{R_2(R_1+ R_2\sin \lambda_L)}{R_2\sin \lambda_L \sin^2 W_L + (R_1+ R_2 \sin \lambda_L)\cos^2 W_L} \qquad (16)$$

En utilisant les équations (15) et (16) , on peut trouver les rayons effectifs du découpage $R(\lambda_L , W_L)_{\psi, Eff, XL=0}$ et $R(\lambda_L, W_L)_{\psi, Eff, ZL=0}$ pour une fraise torique , avec une orientation arbitraire (λ_L, W_L), dans l'usinage des surfaces gauches à cinq axes.

V.5- Orientation et position de l'outil dans le cas de l'usinage à 5 axes :

Dans l'usinage des surfaces gauches à 5 axes, l'interférence est produite quand le rayon de la courbure locale est inférieure au rayon effectif du découpage instantané $R(\lambda_L, W_L)_{\psi, Eff,}$. Pour éviter cette interférence, deux conditions doivent être vérifiées :

$$\left(R(\lambda_L, W_L)_{\psi, Eff, ZL=0} \leq \frac{1}{\rho_{ZL=0}} \right) \text{ et}$$

(17)

$$\left(R(\lambda_L, W_L)_{\psi, Eff, XL=0} \leq \frac{1}{\rho_{XL=0}} \right)$$

Dans la pratique traditionnelle de la programmation de l'usinage à 5 axes , il y a lieu de prédéfinir l'orientation de l'outil (λ_L, W_L)avec quelques petites valeurs des angles($\lambda_{prd,L}$, $W_{prd,L}$),par exemple , on prend $\lambda_{prd,L}=5°$ et $W_{prd,L}=0°$. Les petites valeurs des angles prédéfinies ($\lambda_{prd,L}$, $W_{prd,L}$) ne garantissent pas la vérification des conditions précédentes.

Lorsque l'angle d'inclinaison λ_L est petit, le rayon effectif $R_{\psi, Eff, XL=0}$ devient grand, ce qui peut facilement causer une collision de l'outil avec la surface de pièce si :

$$R_{\psi, Eff, XL=0} > |1/\rho| \qquad (18)$$

Si la condition de l'équation (17) n'est pas satisfaite, les nouveaux angles d'inclinaison λ_L^* ,w^* doivent être recalculés en utilisant l'équation suivante :

$$\lambda_L^* = \sin^{-1}\left(\frac{R_1 R_{2\rho\, ZL=0} - R_1 \cos^2 W_{prd,L}}{R_2(1 - R_{2\rho\, ZL=0})}\right) \qquad (19)$$

$$w^* = \sin^{-1}\left(\frac{P_{XL=0} R_2(R_1 + R_2 \sin\lambda_L^*) - R_2 \sin\lambda_L^*}{R_1}\right)^{1/2} \qquad (20)$$

En utilisant le profil du découpage instantané $W(\theta^*,\Phi^*)_L$ et la courbure de la surface locale pour analyser l'erreur des surfaces usinables,le rayon de la courbure K du profil de découpage $W(\theta^*,\Phi^*)_L$ en point de contact CC^* est calculé comme l'inverse de la courbure ρ en utilisant l'équation (12) avec :

$$K = \frac{1}{\rho_{XL=0}} = \frac{1}{\rho_{min}\cos^2 W_L + \rho_{max}\sin^2 W_L} \qquad (21)$$

Avec :

$$\rho_{\psi,min} = \frac{\sin\lambda_L}{(R_1+R_2)\sin\lambda_L}$$

$$\rho_{\psi,max} = 1/R_2$$

Comme illustrer dans la figure (7), l'outil pénètre la surface avec une erreur négative δ entre P_1 et P_2 et un dégagement Δ entre les points P_3 et P_4. Ces deux types d'erreurs doivent être contrôlées pour être en dessous de la tolérance acceptable. Le centre de la courbure locale est sur le point P_0 où $Y_L = K$ et $Z_L=0$. Le creusage à la gouge δ peut être calculé en utilisant le profil du découpage instantané $W(\theta^*,\Phi^*)^G_L$ présenté dans l'équation (14)., et K est calculé en utilisant l'équation (21) comme suit :

$$\overline{P_0P_2} - \overline{P_0P_1} = \left[(K - y^*_{P2,L})^2 + (Z^*_{P2,L})^2\right]^{1/2} - K$$

$$= \left[(K - y^*_{P2,L})^2 + (Z^*_{P2,L})^2\right]^{1/2} \left(-\frac{1}{\rho_{min}\cos^2 W_L + \rho_{max}\sin^2 W_L}\right) \quad (22)$$

où $y^*_{P2,L}$ et $Z^*_{P2,L}$ sont les éléments y_L et z_L du point P_2 sur le profil du découpage instantané $W(\theta^*,\Phi^*)^G_L$. On peut trouver l'erreur du dégagement Δ avec la même méthode :

$$\overline{P_0P_4} - \overline{P_0P_3} = K\left[(K - y^*_{P3,L})^2 + (Z^*_{P3,L})^2\right]^{1/2}$$

$$= \left(\frac{1}{\rho_{min}\cos^2 W_L + \rho_{max}\sin^2 W_L}\right)\left[(K - y^*_{P3,L})^2 + (Z^*_{P3,L})^2\right]^{1/2} \quad (23)$$

Dans l'équation (23), $y^*_{P3,L}$, $Z^*_{P3,L}$ sont les éléments y_L et z_L du point P_3 sur le profil du découpage instantané $W(\theta^*,\Phi^*)^G_L$. Pour une tolérance de surface donnée T_{Tol}, les deux erreurs Δ et δ doivent être commandées pour satisfaire les deux conditions suivantes :

$$\delta \le T_{Tol} \quad \text{et} \quad \Delta \le T_{Tol} \quad (24)$$

Dans la génération de la trajectoire d'outil à cinq axes l'orientation de l'outil (λ_L, W_L) doit être choisie pour maintenir δ et Δ dans la marge de la tolérance T_{Tol} afin d'éviter des erreurs inacceptables de surface de pièces.

V.6- Calcul des coordonnées de la position de l'outil :

Pour générer le code d'usinage de la commande numérique, la position du bout de l'outil doit être calculée (CL). Le point de contact de l'outil est habituellement le premier déterminé en se basant sur la géométrie de la surface. Comme présenté dans la figure 14 , la position du bout de l'outil CL^*_M (Cutter Location), est calculée par l'éxcentrage du point de contact CC^*_M avec un vecteur $\vec{V}$ en utilisant la procédure suivante :

$$\vec{CL}^*_M = \vec{CC}^*_M + \vec{V} \qquad (25)$$

Où $\vec{V}$ est la différence entre le bout de l'outil CL^*_M et le point de contact de l'outil CC^*_M.

De l'équation (4) on peut tirer l'équation suivante:

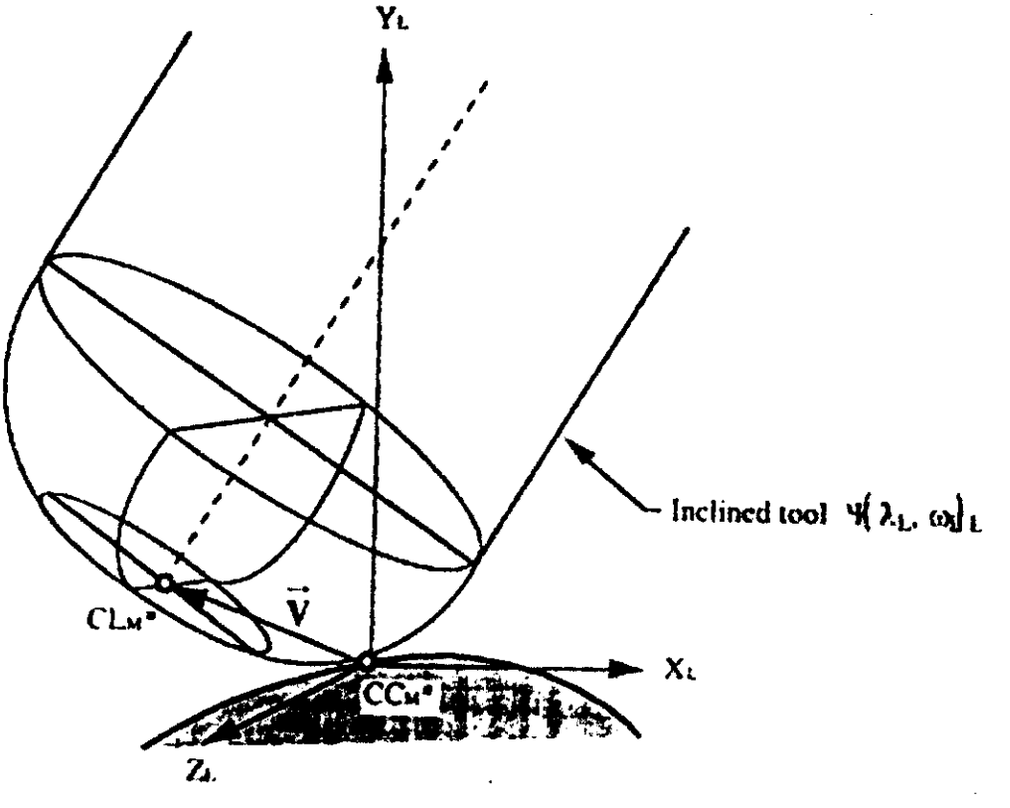


Figure III.12 : *La position du bout de l'outil en fonction du point de contact de l'outil CC_M*

$$\vec{V} = \begin{pmatrix} -R\cos w_L \cos\lambda_L - R_2 \cos w_L \sin\lambda_L \\ R_2 + R_1 \sin\lambda_L - r_2 \cos\lambda_L \\ -R_1 \sin w_L \cos\lambda_L - R_2 \sin w_L \sin\lambda_L \end{pmatrix} \quad (26)$$

La position du bout de l'outil CL_M (X_L^{CL}, Y_L^{CL}, Z_L^{CL}) peut être trouvée d'après le point de contact de l'outil (CC), CC_M(X_L^{CC}, Y_L^{CC}, Z_L^{CC}) sur la surface de la pièce comme montré ci-dessous :

$$\begin{pmatrix} X_L^{CL} \\ Y_L^{CL} \\ Z_L^{CL} \end{pmatrix}_L = \begin{pmatrix} X_L^{CC} \\ Y_L^{CC} \\ Z_L^{CC} \end{pmatrix}_L + \vec{V} = \begin{pmatrix} X_L^{CC} - R\cos w_L \cos\lambda_L - R_2 \cos w_L \sin\lambda_L \\ Y_L^{CC} + R_2 + R_1 \sin\lambda_L - R_2 \cos\lambda_L \\ Z_L^{CC} - R_1 \sin w_L \cos\lambda_L - R_2 \sin w_L \sin\lambda_L \end{pmatrix}_L \quad (27)$$

En utilisant l'équation (27), la position de l'outil peut être facilement calculée en fonction du point de contact de l'outil (CC). Dans l'usinage à cinq axes, la position de l'outil (CL) et l'orientation de l'outil (λ_L, W_L) sont utilisées pour générer le programme de l'usinage.

VI- ETUDE COMPARATIVE ENTRE L'USINAGE 3 AXES et 5 AXES[10]

L'usinage des pièces de forme complexes est réalisé à l'aide d'outils à bout sphérique généralement soit sur une machine à 3 axes soit 5 axes. Généralement la trajectoire est décomposée en segments de droites traités par l interpolation linéaire de la CN.

Pour les machines 3 axes on pilote la pointe de l'outil .Pour une surface usinable il existe un point unique de tangence entre l'outil et la pièce,ainsi suivant la forme de la surface, la vitesse de coupe locale varie à toute instant. Il se pose également souvent des problèmes d'accessibilité de la surface à réaliser ce qui conduit à des empilements de rallonge et d'attachement pour monter l'outil.

Lorsque l'on utilise une machine à 5 axes, on pilote à la fois la pointe et l'orientation de l'axe de l'outil. Les possibilités offertes par les deux axes de rotation supplémentaires permettent l'optimisation de l'orientation de l'outil en fonction de critère tels que la vitesse de coupe constante, le contrôle de l'épaisseur de copeau, l'augmentation de la zone usinée, la collision de l'outil avec son environnement .

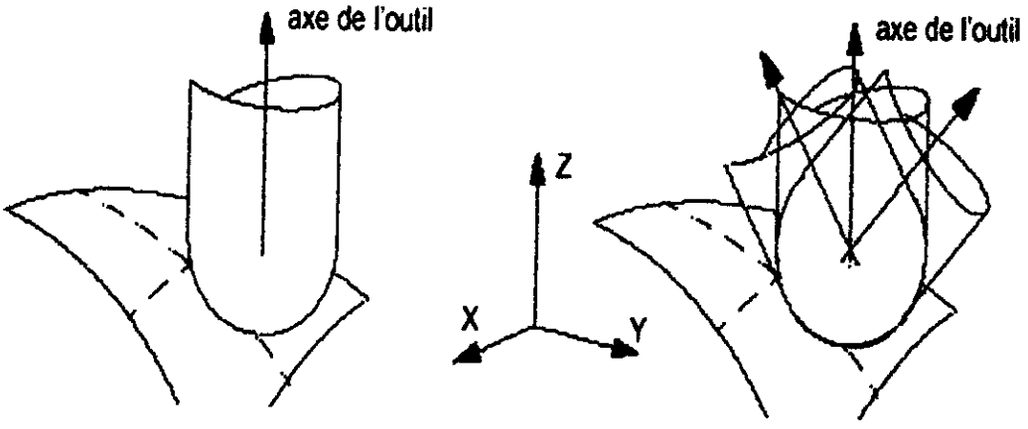


***Figure III.13** : Comparaison entre l'usinage à 3 axes et l'usinage à 5axes*

VII- Analyse d'usinabilité d'une surface :[12]

L'analyse d'une usinabilité consiste à appliquer sur la surface un raisonnement géométrique permettant d'identifier si la surface satisfait l'ensemble des contraintes technologiques du procédé. Cette analyse permet d'enrichir le modèle surface par des informations géométrique relatives aux ressources comme les machines pouvant être utilisées, les posages potentiels sur la machine , les outils potentiellement utilisables.
Le raisonnement géométrique appliqué concerne quatre aspects de l'usinabilité : *la visibilité et l'accessibilité topologique* de la surface, *la visibilité et l'accessibilité technologique* de la surface. La visibilité et l'accessibilité topologique de la surface dépendent de contraintes purement géométriques. Il n'est pas tenu du procédé dans cette analyse. La visibilité et l'accessibilité technologique de la surface dépendent de contraintes à la fois géométriques et technologiques. Le type d'outil choisi ainsi que le procédé sont pris en compte de l'usinabilité.

VII.1- Visibilité topologique

La visibilité d'une surface est l'ensemble des vecteurs directeur des droites de vision de la surface. La visibilité de la surface dépend de sa géométrie : plan, cylindre ou sphère.

La visibilité de l'entité est caractérisée par une carte de visibilité Θ_{map} située sur une sphère de Gauss. Cette carte est l'ensemble des vecteurs directeur des droites pour lesquelles toutes les faces de l'entité sont visibles. Elle est définie à partir des vecteurs normaux à chacune des faces de l'entité orienté vers l'extérieur de la matière. La définition de cette carte de visibilité est la suivante :

VII.2- Accessibilité topologique

L'accessibilité topologique caractérise l'aptitude d'un outil à atteindre un ensemble de points caractéristiques d'une entité de forme sans couper les autres faces de la pièce.

VII.3- Visibilité technologique

La visibilité technologique caractérise l'aptitude d'un outil à « voir » l'entité. Elle dépend de l'orientation des outils dans l'espace de travail de la machine .Cette orientation correspond à l'axe de rotation de l'outil.

VII.4- Accessibilité technologique

L'accessibilité technologique caractérise la non interférence de l'outil avec l'entité lors de l'usinage. Elle tient compte des dimensions de l'outil de coupe et des trajectoires de l'outil pendant l'usinage.

Par exemple, pour ébaucher une poche, il existe un diamètre maximal d'outil au-delà duquel l'entité ne pourra pas être usiné. La définition de ce diamètre consiste à rechercher le plus grand diamètre inscrit à l'intérieur du contour de la poche.

VIII- LES INTERFERENCES [13]

Sur les fraiseuses à 3 axes le problème d'interférence est du seulement à la géométrie de la surface et de l'outil. Il existent 4 cas (scénario) d'interférence :

- Le rayon d'une courbure locale de la surface est inférieur au rayon de l'outil.

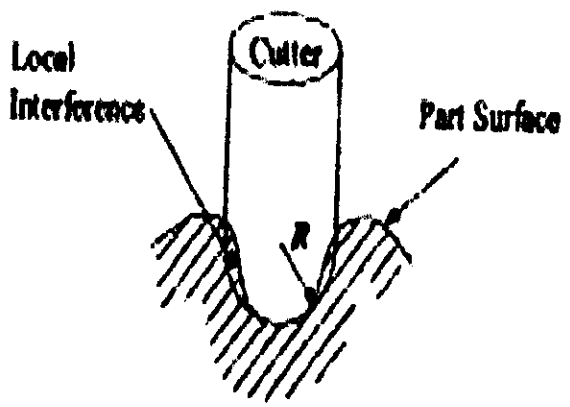


Figure III.14 : Interference locale

- Il existent des parties de la surface ou l'angle entre la normale de la surface et l'axe de l'outil est supérieur à 90° (Protrusion Interference **PI**)

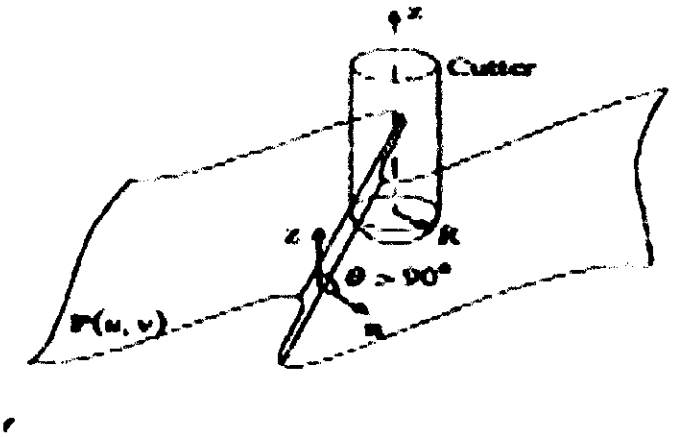


***Figure III.15** : Protrusion Interference*

- Certaines sous surfaces ne sont pas visibles dans la direction de l'axe de l'outil (Overlapping Interference **OI**).

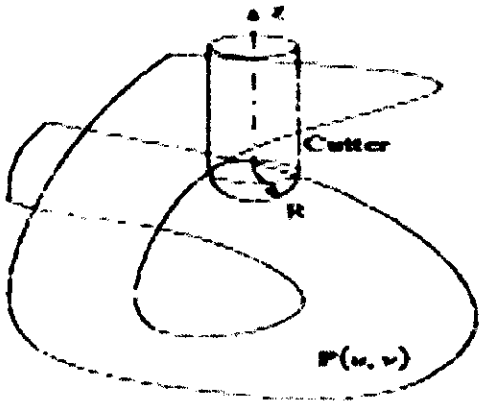


***figure III.16** : Overlapping Interference*

- L'outil entre en collision avec les frontières de la surface (**Boundary Collision** Interference **BCI**) .

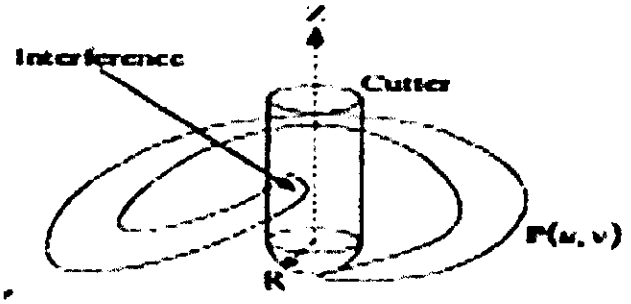


***Figure III.17** : Boundary collision Interference*

IX- L'USINABILITE DES SURFACES GAUCHES SUR LES FRAISEUSES A 3 AXES

Beaucoup d'algorithmes sont développés pour enlever les sous surfaces cachées d'une surface. Il est évident qu'une fraiseuse 3 axes ne peut avoir l'accès à des parties de la surface, qui se trouvent cachées. L'algorithme d'enlèvement des surfaces cachées peut presque directement être employé pour décider quelles parties(visibles) de la surface peuvent être usinées sur une fraiseuse 3 axes, et il donne l'orientation de la direction de l'outil.

Pour usiner une surface gauche, il faut que l'outil reste constamment tangent à la surface en tout point de contact avec la surface à usiner.

Dans le cas de l'usinage sur des fraiseuses à 3 axes, les fraises à bouts sphériques sont les plus utilisées. Dans la fraise boule, on peut calculer facilement la trajectoire, car il suffit que le centre *C* de la demi sphère ; extrémité de la fraise ; se déplace sur la surface (*S'*) parallèlement à la surface (*S*) à obtenir à une distance *R* pour qu'un point *P* de la sphère glisse sur (*S*). On remarque qu'il faut que l'angle α reste inférieur à 90° entre l'axe de l'outil et la normale $\vec{n}$ à la surface (*S*) en *P* dans le cas de l'usinage sur une machine à trois axes.

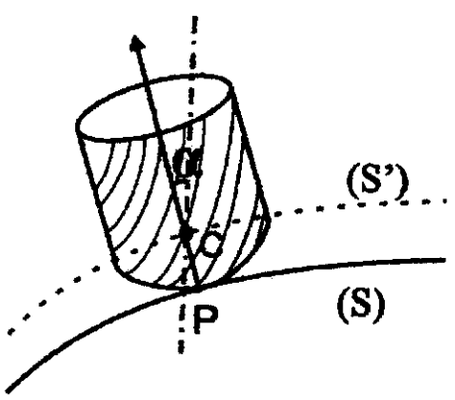


Figure III.18 : *Déplacement du demi sphere de l'outil sphérique*

Un inconvénient se pose lorsque le point *P* s'approche de l'axe de la fraise tel que α est petit. En effet, dans cette zone la vitesse de coupe V devient très faible et l'usinage se fait dans de mauvaises conditions.

La vitesse de coupe est donnée par: $V = \alpha.a.N$.

Avec: *N*: est la fréquence de rotation de la broche.

a: est le diamètre de la partie de la fraise en contact avec la surface.

Pour avoir de bonnes conditions d'usinage, deux solutions sont possibles :

Première solution : on utilise une fraise torique dont la partie active reste toujours éloignée de l'axe de l'outil.

Deuxième solution : on utilise une fraiseuse avec plus de degrés de liberté (4 ou 5 axes) qui imposent un angle α minimum, ou constant entre l'axe et la normale $\vec{n}$ à la surface.

X- L'USINABILITE D'UNE SURFACE GAUCHE SUR LES MACHINES A 5 AXES [14]

En usinage à trois axes, pour une surface usinable, il existe une unique position de l'outil tangente à la surface en une position donnée. Par contre, en usinage à cinq axes, il existe une infinité de positions de l'outil tangentes à la surface au point de contact. Ces degrés de liberté supplémentaires rendent le problème sous contraint lors du calcul du trajet d'usinage, mais permettent l'optimisation de l'orientation de l'outil en fonction de critères tels que la minimisation des efforts de coupe ou l'augmentation de la zone usinée.

L'usinage à cinq axes permet de réaliser des surfaces en ayant un seul point de contact (usinage en bout) ou une ligne (usinage par flanc) de contact entre l'outil et la surface nominale. Dans le cadre de l'usinage en bout, la surface est réalisée par balayage de l'outil.

Les libertés d'orientation de l'outil permettent :

- d'usiner des surfaces en contre-dépouille, par orientation de l'axe de l'outil de façon à éliminer les collisions entre l'outil et la surface ;

- d'optimiser la vitesse de coupe dans le cas de l'utilisation d'un outil hémisphérique; par rotation de l'axe de l'outil, le point de contact court le long de l'arête de coupe et la valeur du rayon effectif de coupe varie ;

- de changer de géométrie d'outil. Cette technique autorise l'usage d'outils toriques, sans risque de talonnement du fond de l'outil sur la surface. Il est alors possible d'usiner des petits rayons de raccordement, dont la valeur est supérieure à celle du petit rayon du tore, avec un outil plus massif et plus rigide. La valeur du grand rayon du tore permet de couvrir la surface en moins de balayages et donc en moins de temps.

Dans le cadre de l'usinage avec le flanc de l'outil, le contact entre l'outil et la surface est une ligne le long de l'outil. Ici toute la partie coupante est en contact avec la surface. Cette technique ne peut être utilisée que pour des types de surfaces particulières. Les surfaces les plus simples à usiner avec des flancs d'outil rectilignes sont les surfaces développables. Les autres surfaces peuvent être usinées par cette technique, si on autorise une certaine erreur au niveau du contact entre la surface et l'outil.

Par contre, l'ajout de deux angles d'orientation pose un problème à la construction de la trajectoire d'usinage. La recherche de l'orientation de l'outil est un problème d'optimisation difficile à mettre en oeuvre pour des surfaces ayant de fortes variations de courbure. Les risques de collision sont importants. Il est nécessaire de tester le risque de collision sur l'ensemble de l'outil, partie coupante et corps d'outil.

XI- CONCLUSION

Après avoir étudié dans les chapitres précédents séparément les surfaces gauches et les machines outils à commande numérique,nous avons étudié, dans ce chapitre l'usinage des surfaces complexes sur les fraiseuses numériques à trois axes et à cinq axes . Nous avons aussi analysé l'usinabilité des surfaces gauches sur les fraiseuses numériques en donnant les conditions d'usinabilité.

La maitrise de ces thèmes nous permet de passer à l'étude de la partie principale de notre projet qui consiste à la vérification de l'usinabilité des surfaces complexes .

CONCEPTION ET REALISATION DU SYSTEME

4

DANS CE CHAPITRE

- PROBLEMATIQUE ET SOLUTION PROPOSEE
- DEFINITION DES BESOINS
- LA VUE STATIQUE DU SYSTEME
- LA VUE DYNAMIQUE DU SYSTEME

I- INTRODUCTION

Dans cette partie nous allons présenter l'approche utilisée pour résoudre le problème de l'usinabilité des surfaces gauches sur des fraiseuses numériques à trois axes et à cinq axes.

Vu la complexité des conditions de l'usinabilité à cinq axes à savoir ; les difficultés d'implémentation des algorithmes de l'usinabilité et de détection des éventuelles collisions ; d'une part , et d'autre part le temps imparti à l'élaboration de notre mémoire n'étant pas suffisant nous nous sommes limités à l'étude de l'usinabilité des surfaces gauches sur des fraiseuses à trois axes.

La conception est la transformation de notre compréhension des conditions requises en un modèle pouvant ensuite être implémenté sous la forme d'un logiciel. Un document de conception est produit à la fin de ce processus

Un document de conception est divisé en deux sections : ***la conception des classes*** et ***les mécanismes architecturaux***.

La section Conception des classes est , à son tour, divisée en deux sous parties :la conception statique(qui détaille les diverses classes avec leurs relations et caractéristiques) et la conception dynamique (qui décrit la façon dont les classes interagissent).

La section des Mécanismes architecturaux fournit tous les détails concernant la façon dont nous allons implémenter la persistance des objets , les accès concurrents , un système d'objets distribués...etc

Dans la pratique nous avons suivi la méthode de conception XP[(1)], méthode itérative et incrémentale. Pour présenter notre système on a choisi le langage UML[(2)]. Il s'articule autour de neuf diagrammes différents, représentant le système selon deux modes : l'un concernant sa structure et l'autre sa dynamique de fonctionnement.

Pour modéliser et concevoir notre système on a utilisé trois[1] des neufs diagrammes:

- Les cas d'utilisation (use case), pour définir les besoins de notre système.
- Le diagramme des classes, pour représenter son architecture (vue statique).
- Le diagramme de séquence, pour représenter les interactions entre les objets (vue dynamique).

[(1)]Voir Annexe B
[(2)] Voir Annexe A

II- PROBLEMATIQUE ET SOLUTION PROPOSEE

Nous avons vu précédemment que les surfaces gauches ne peuvent pas être usinées sur des machines conventionnelles, pour cela on utilise des machines à commande numérique pour les usiner .

Pour vérifier l'usinabilité des surfaces complexes et pour éviter la production de collisions qui risquent d'endommager la pièce et la machine, on a naturellement recours à la simulation. En effet simuler l'expérience sensible de l'objet, c'est-à-dire imaginer l'utilisation ou le comportement, constituent un mode d'anticipation de l'objet à produire et donc, dans une certaine mesure, sa validation. La pré-production de l'objet par simulation constitue en somme une approximation de la solution réelle.

II.1- Définition de la simulation :

La simulation est le processus de conception de modèles d'un système réel, et de conduite d'expérience dont le but est ; soit de comprendre le comportement de système, soit d'évaluer différentes stratégies pour son fonctionnement (dans la limite imposée par un critère ou un ensemble de critères). Elle représente une approche forte appréciée notamment dans la conception.

II.2- Pourquoi simuler dans le cas de la production ?

L'utilisation de la commande numérique sur les machines-outils (MO) distancie l'homme de la matière travaillée par la machine. Le rapport avec la machine n'est plus physique comme sur les machines conventionnelles. Avec les MOCN, ce rapport est médiatisé par l'informatique (la commande numérique) et l'homme n'intervient plus que sur un clavier, en contrôlant ses ordres sur un écran. La capacité d'abstraction requise est d'autant plus grande que les ordres adressés à la machine ne le sont pas pour une exécution immédiate, mais pour une exécution différée avec un programme

La médiatisation par l'informatique du rapport homme / matière est intensifiée sur les ensembles complexes intégrant de multiple fonctions (lignes continues de fabrication, ateliers flexibles). D'une part, les installations sont de plus en plus grandes avec des systèmes de sécurité qui éloignent les opérateurs du processus de fabrication donc de la matière proprement dite. D'autre part, les asservissements informatiques des différents composants sont de plus en plus éloignés des machines elles même et arrangées dans des configurations totalement étrangères à celles des machines.

Les opérateurs, y compris les opérateurs de base, doivent se construire une image, une représentation juste et efficace du système total .En plus, la surveillance le contrôle et le dépannage requièrent une puissante capacité de diagnostic. Les pannes et les arrêts peuvent venir de multiples causes. On arrive ainsi au paradoxe suivant : plus les systèmes sont complexes, plus la probabilité de pannes est grande (fragilisation de système complexe), plus l'opérateur est éloigné du processus lui-même , c'est-à-dire qu'il doit se constituer à distance une image non réductrice de l'objet réel complexe

La nécessité de simuler

a) est utile pour vérifier un programme avant son exécution, en remédiant aux erreurs et en palliant aux risques de collision ;

b) est indispensable pour évaluer avec précision les temps d'exécution en vue d'élaborer des devis précis.

c) Est nécessaire pour optimiser le choix entre plusieurs procédures d'exécution et ainsi parvenir aux choix économiques garantissant la qualité exigée.

Pour qu'une simulation fournisse les résultats escomptés, il est nécessaire de connaître les principes directeurs mis en œuvre et les règles qui ont conduit aux objectifs proposés, c'est-à-dire une gamme de production opérationnelle.

III- DEFINITION DES BESOINS

Dans cette deuxième étape, nous utilisons le diagramme des cas d'utilisation. Le résultat de cette étape est une description sommaire de ce que doit faire notre système.
Dans le cas de notre étude, le diagramme des cas d'utilisations s'organise en quatre modules complémentaires, ceux introduits ci-dessous (figure IV.1).

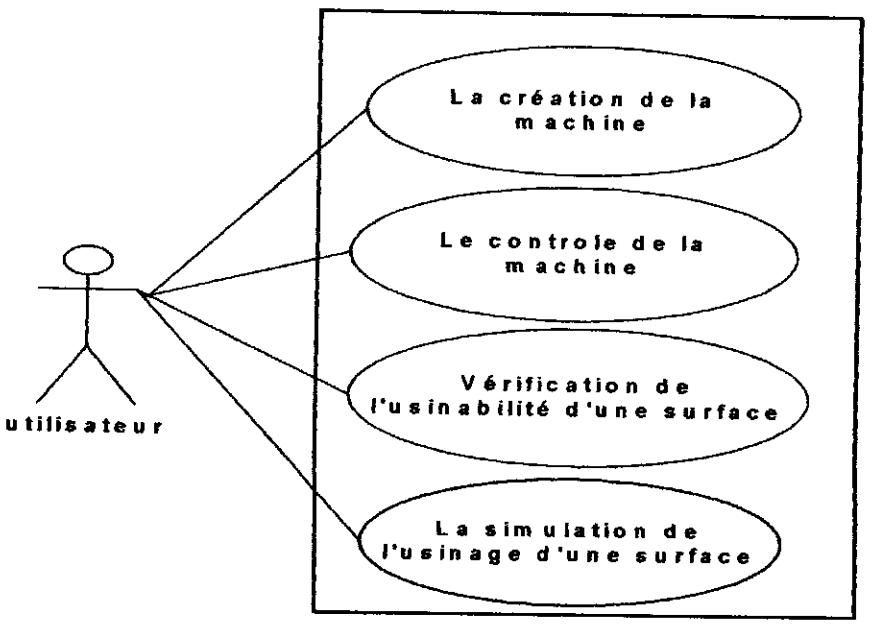


Figure IV.1 *:Les modules du système*

Notre système permet à l'utilisateur de :

- créer une machine,
- contrôler cette machine,
- vérifier l'usinabilité d'une surface donnée sur cette machine.
- Simuler l'usinage de cette surface sur la machine

Dans la suite, on présentera les cas d'utilisation correspondants à ces modules :

- ***La création de la machine***

Pour la création de la machine, il faut :

- donner le nombre d'axes de la machine à construire (exemple : machine à 3 axes, machine à 4 axes , machine à 5 axe).
- donner les paramètres de la machine (les dimensions de la table, les dimensions de la broche,les dimensions de l'outil et les courses de la machine)
- choisir la configuration de la machine (exemple : les axes de translation X,Y pour la table , l'axe de déplacement Z pour la broche).

Le diagramme suivant montre en détails le module de la création de la machine (figure IV.2).

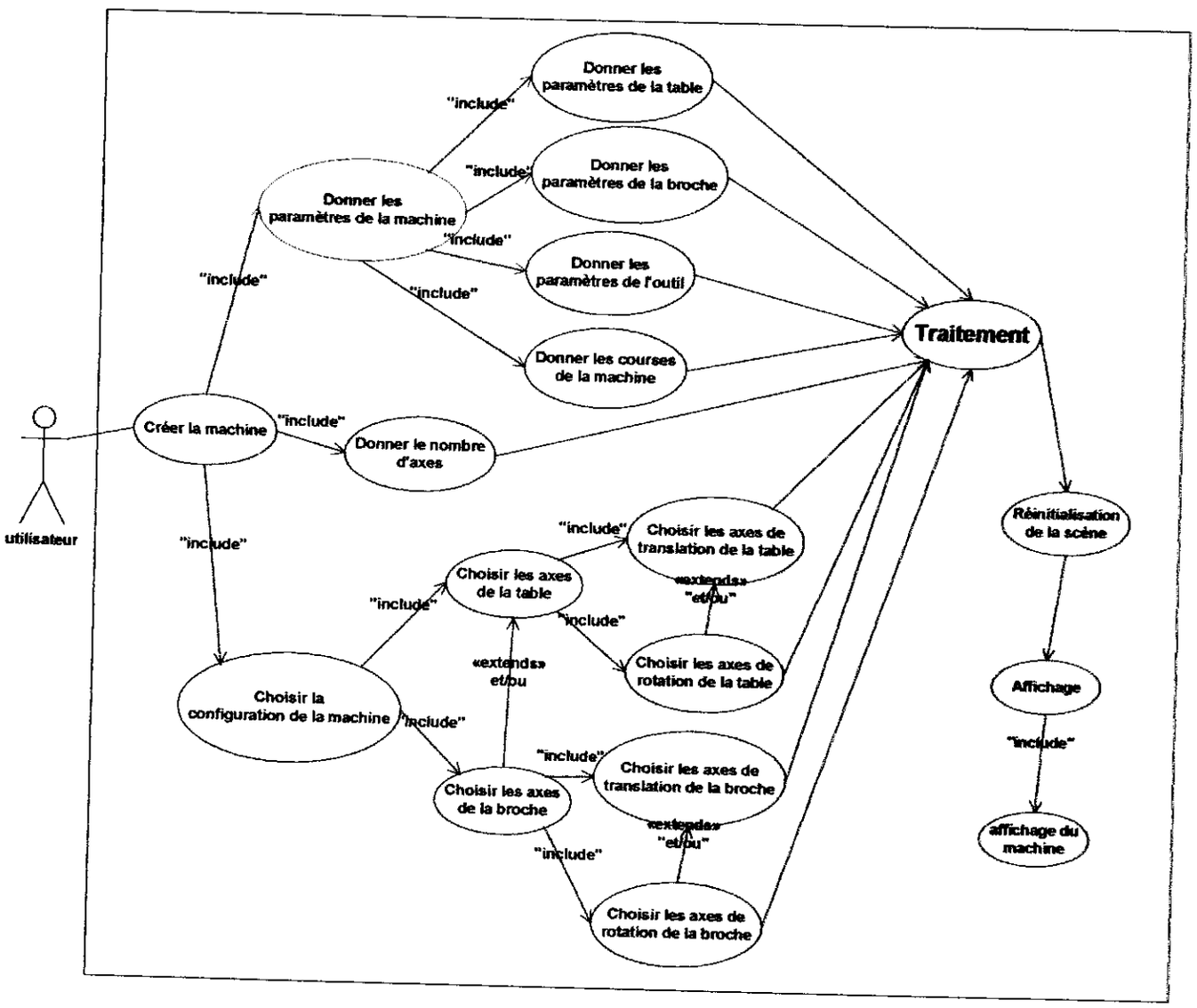


figure IV.2 *:Le diagramme des cas d'utilisation du module création de la machine*

- ***Contrôle de la machine :***

Le diagramme de la ***figure VI.3*** présente les cas d'utilisation correspondant à ce module.

Le système permet à l'utilisateur de :

- Contrôler le déplacement de la table ;
- Contrôler le déplacement de la broche ;
- Contrôler la camera.

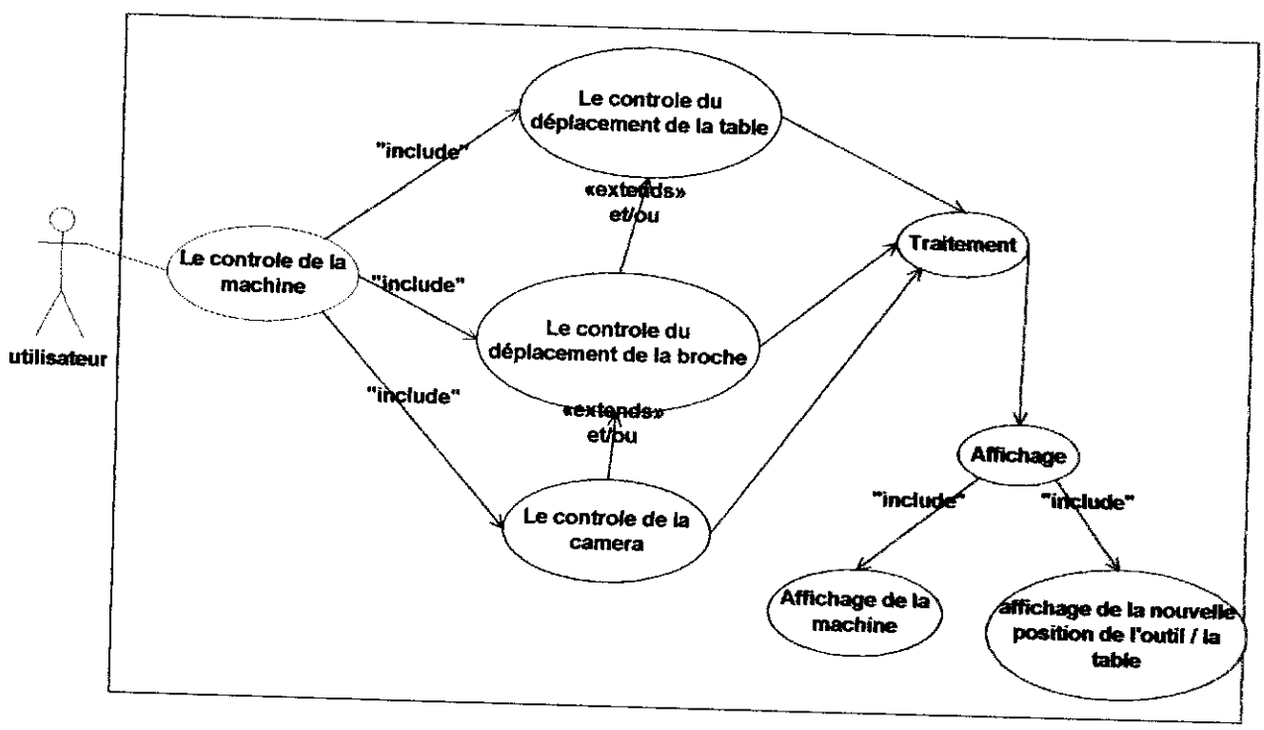


figure IV.3 : *le diagramme des cas d'utilisation du module Control de la machine*

- ***Vérification de l'usinabilité d'une surface donnée***

Pour vérifier l'usinabilité d'une surface donner , il faut :

- Entrer la surface (un nuage de points issu de la digitalisation de la surface à usiner)

Notre système nous permet de faire le maillage de la surface et en appliquant les notions de l'accessibilité et la visibilité[12] , il permet à la fin d'identifier les zones de la surface non visibles et les zones non accessible .

[1] voir chapitre 3

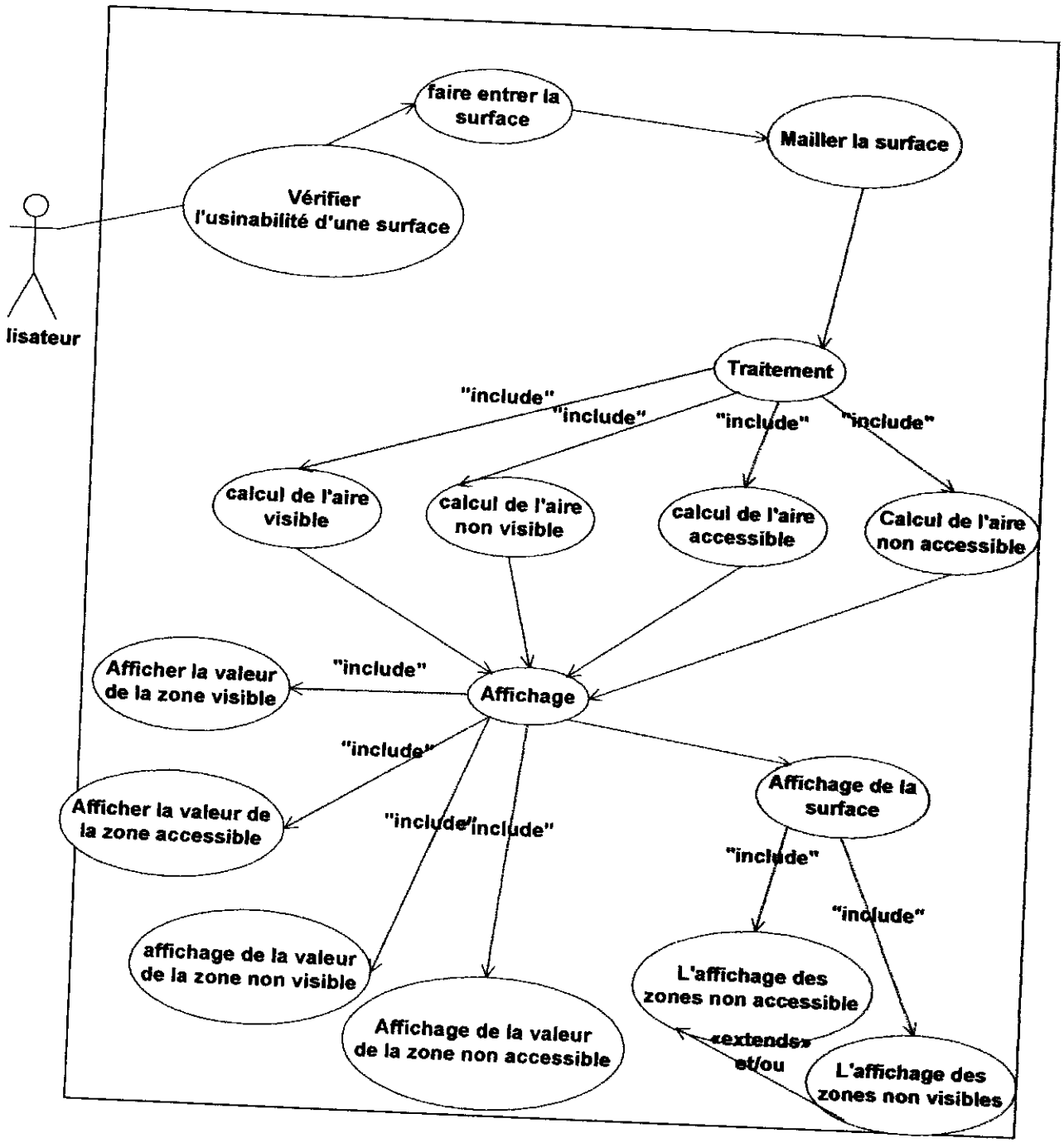


***Figure IV.4** : le digramme des cas d'utilisation du module vérification de l'usinabilité*

- **La simulation de l'usinage**

pour simuler l'usinage , il faut :

- faire entrer le brut à usiner soit en donnant les dimensions de brut ou bien en utilisant les dimensions de brut minimal.
- Faire entrer la trajectoire de l'outil (l'ensemble des positions suivies par l'outil).

Et notre système permet de simuler l'usinage tout en visualisant les différents mouvements des deux organes mobiles de la machine

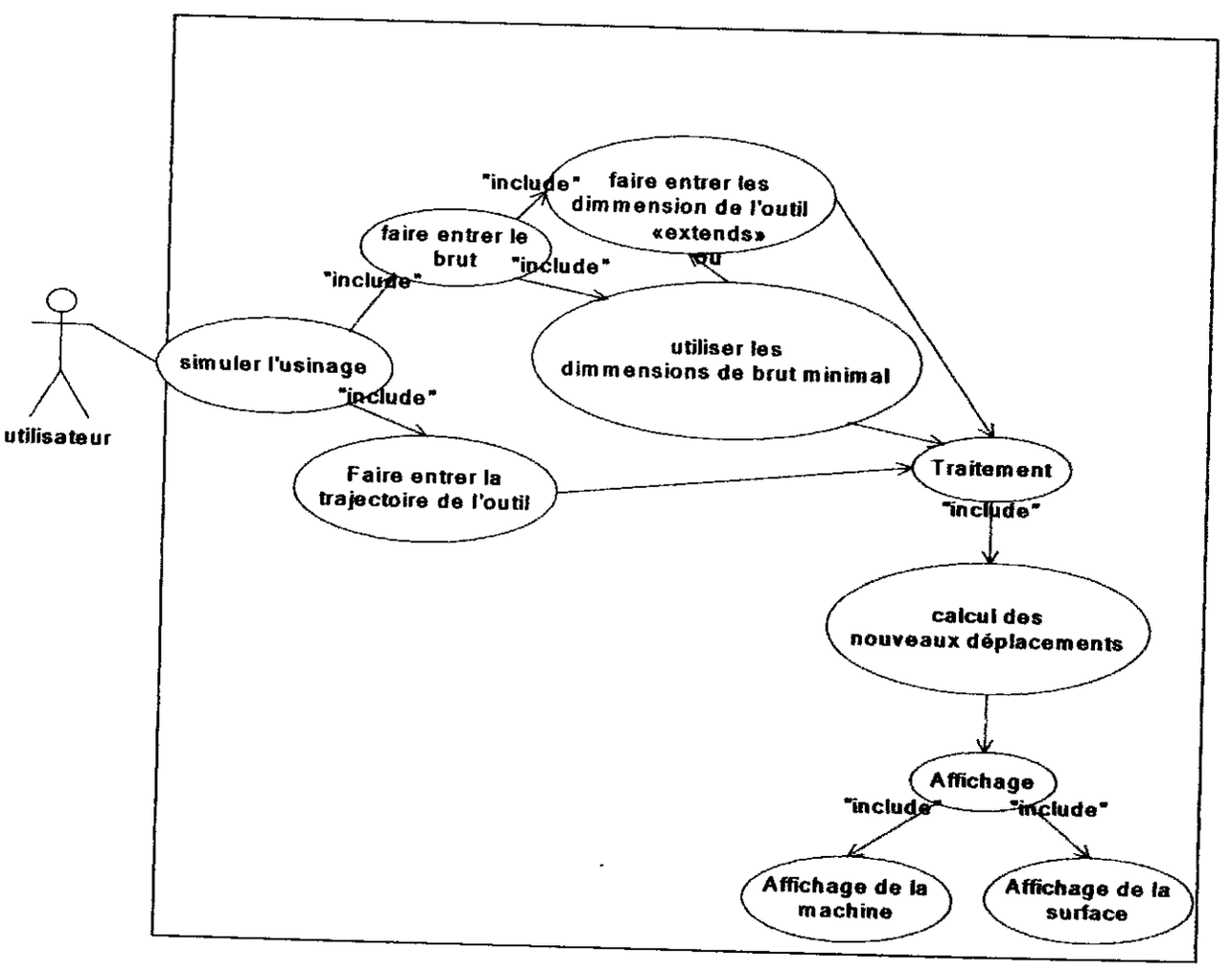


figure IV.5 : *le diagramme des cas d'utilisation du module de la simulation de l'usinage*

IV- LA VUE STATIQUE DU SYSTEME

IV.1- Architecture du système :

Le résultat de cette étape est une description générale de l'architecture du système permettant d'identifier les différents composants et leurs liens.

Dans cette étape, nous utilisons le diagramme des classes . Ce dernier nous permet de modéliser, d'une manière statique, les relations qui existent entre l'ensemble de classes. Il développe d'une part la structure des entités du système et d'autre part celle d'un code orienté objet.

Afin de permettre au lecteur de bien comprendre notre système, nous présentons dans ce qui suit une vue d'ensemble sur l'architecture du système. Ensuite nous allons oeuvrer à décomposer et à détailler chaque composant à part.

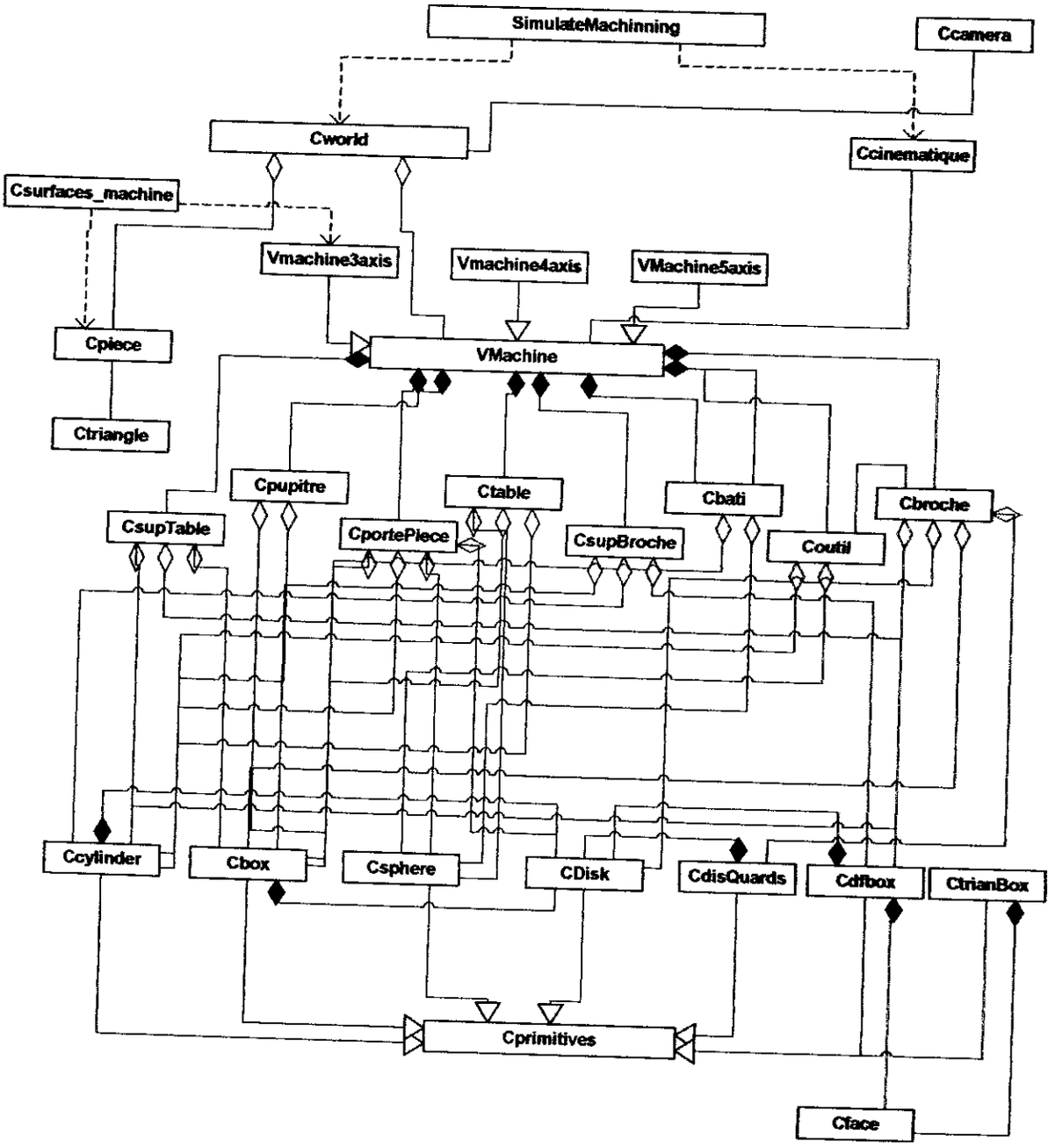


Figure IV.6 : *Architecture du système*

IV.2- La gestion de la scène :

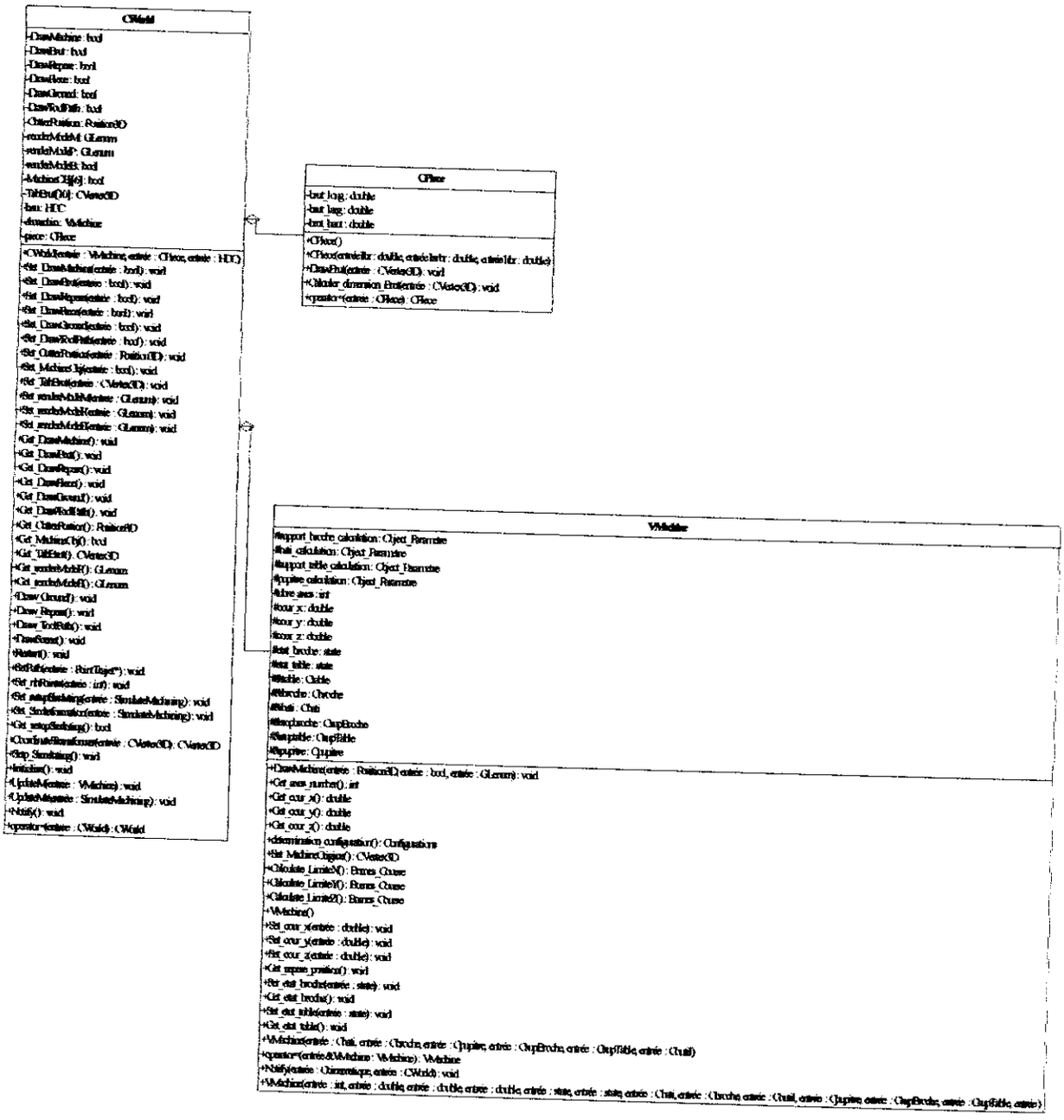


figure IV.7 *: la gestion de la scène*

- **La classe Cworld :**

Dans cette section, on va détailler l'architecture de notre scène , qui est gérée par la classe Cworld . Cette dernière doit avoir une référence aux deux classes Vmachine qui implémente notre machine et Cpiece qui présente la pièce à usiner.

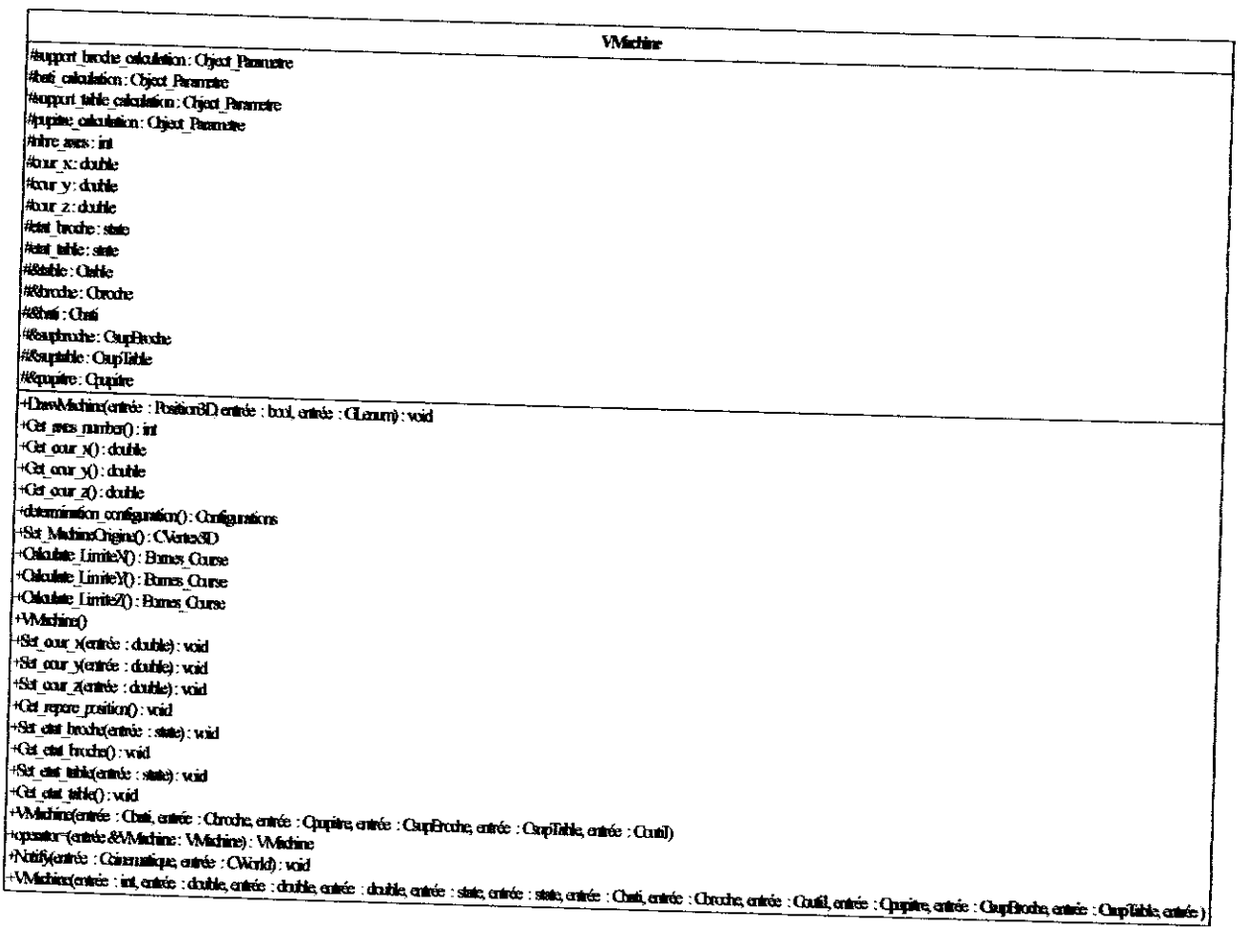


figure IV.8 : *la classe CWorld*

IV.3- Architecture de la machine :

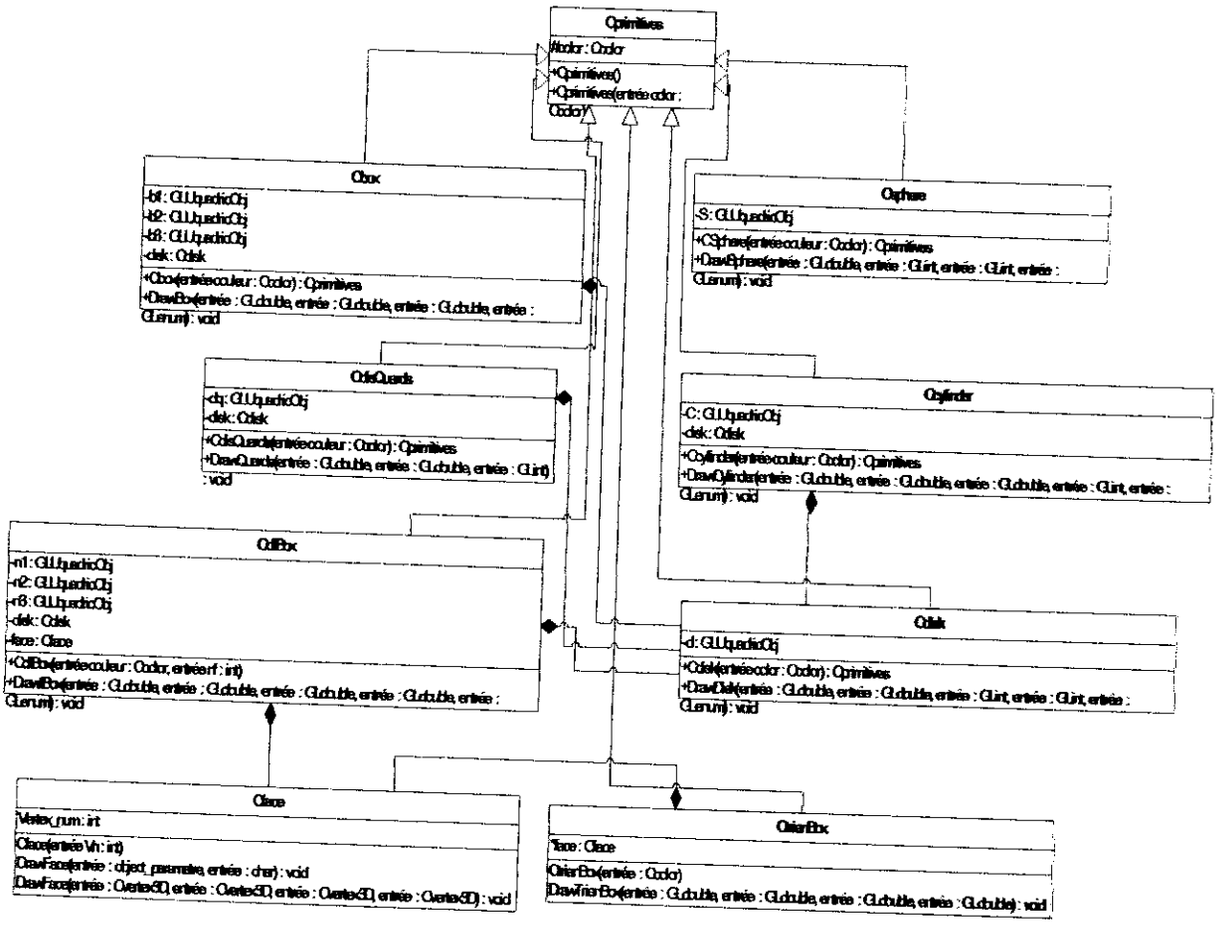
- **La Classe Cprimitives :**

La question qui peut se poser est comment représenter la géométrie de notre machine (fraiseuse) ?. Pour répondre à cette question, on utilise une classe pour chaque partie géométrique de la machine , par exemple : classe Bâti, classe broche, classe table,...etc

Cprimitives
#color : Ccolor
+Cprimitives() +Cprimitives(entrée color : Ccolor)

Figure IV.8 : *la classe Cprimitives*

Chaque classe permet la construction de la partie correspondante grâce à sa méthode : DrawPartie (*DrawBati, DrawBroche, DrawTable*...) et ce, par l'envoi de message à l'une ou plusieurs sous classes de la classe ***Cprimitives***. Ces sous-classes permettent la création (construction géométrique) des objets simples (sphère , cylindre, box...).



la figure IV.9 :*La classe Cprimitives et ses sous classes*

- ***La classe Cbox :*** Cette classe permet la construction d'un parallélépipède par sa méthode DrawBox().

Cbox
-b1 : GLUquadricObj -b2 : GLUquadricObj -b3 : GLUquadricObj -disk : Cdisk
+Cbox(entrée couleur : Ccolor) : Cprimitives +DrawBox(entrée : GLdouble, entrée : GLdouble, entrée : GLdouble, entrée : GLenum) : void

Figure IV.10 *: la classe Cbox*

- ***La classe Cdisk :*** Cette classe permet la construction d'un disque par sa méthode DrawDisk().

Cdisk
-d : GLUquadricObj
+Cdisk(entrée color : Ccolor) : Cprimitives +DrawDisk(entrée : GLdouble, entrée : GLdouble, entrée : GLint, entrée : GLint, entrée : GLenum) : void

figure IV.11 *: la classe Cdisk*

- ***La classe CdfBox :*** Cette classe permet la construction d'un box déformé par sa méthode DrawdfBox().

CdfBox
-n1 : GLUquadricObj -n2 : GLUquadricObj -n3 : GLUquadricObj -disk : Cdisk -face : Cface
+CdfBox(entrée couleur : Ccolor, entrée nf : int) +DrawfBox(entrée : GLdouble, entrée : GLdouble, entrée : GLdouble, entrée : GLdouble, entrée : GLenum) : void

figure IV.12 *: la classe CdfBox*

- ***La classe CdisQuards :*** Cette classe permet la construction d'un disque carré par sa méthode DrawQuards().

CdisQuards
-dq : GLUquadricObj -disk : Cdisk
+CdisQuards(entrée couleur : Ccolor) : Cprimitives +DrawQuards(entrée : GLdouble, entrée : GLdouble, entrée : GLint) : void

***figure IV.13** : la classe CdisQuards*

- ***La classe Ccylinder :*** Cette classe permet la construction d'un cylindre par sa méthode DrawCylinder().

Ccylinder
-C : GLUquadricObj -disk : Cdisk
+Ccylinder(entrée couleur : Ccolor) : Cprimitives +DrawCylinder(entrée : GLdouble, entrée : GLdouble, entrée : GLdouble, entrée : GLint, entrée : GLenum) : void

***figure IV.14** : la classe Ccylinder*

- ***La classe Csphere :*** Cette classe permet la construction d'une sphère par sa méthode DrawSphere().

Csphere
-S : GLUquadricObj
+CSphere(entrée couleur : Ccolor) : Cprimitives +DrawSphere(entrée : GLdouble, entrée : GLint, entrée : GLint, entrée : GLenum) : void

***figure IV.15** : la classe Csphere*

- **La Classe VMachine:**

Le diagramme suivant représente la classe Vmachine et ses sous classe Vmachine3axis, Vmachine4axis et Vmachine5axis. Ces sous _classes sont héritées de la classe générale Vmachine. Chacune de ces classe représente un type de machine :les machines à trois axes, les machines à quatre axes et les machines à cinq axes

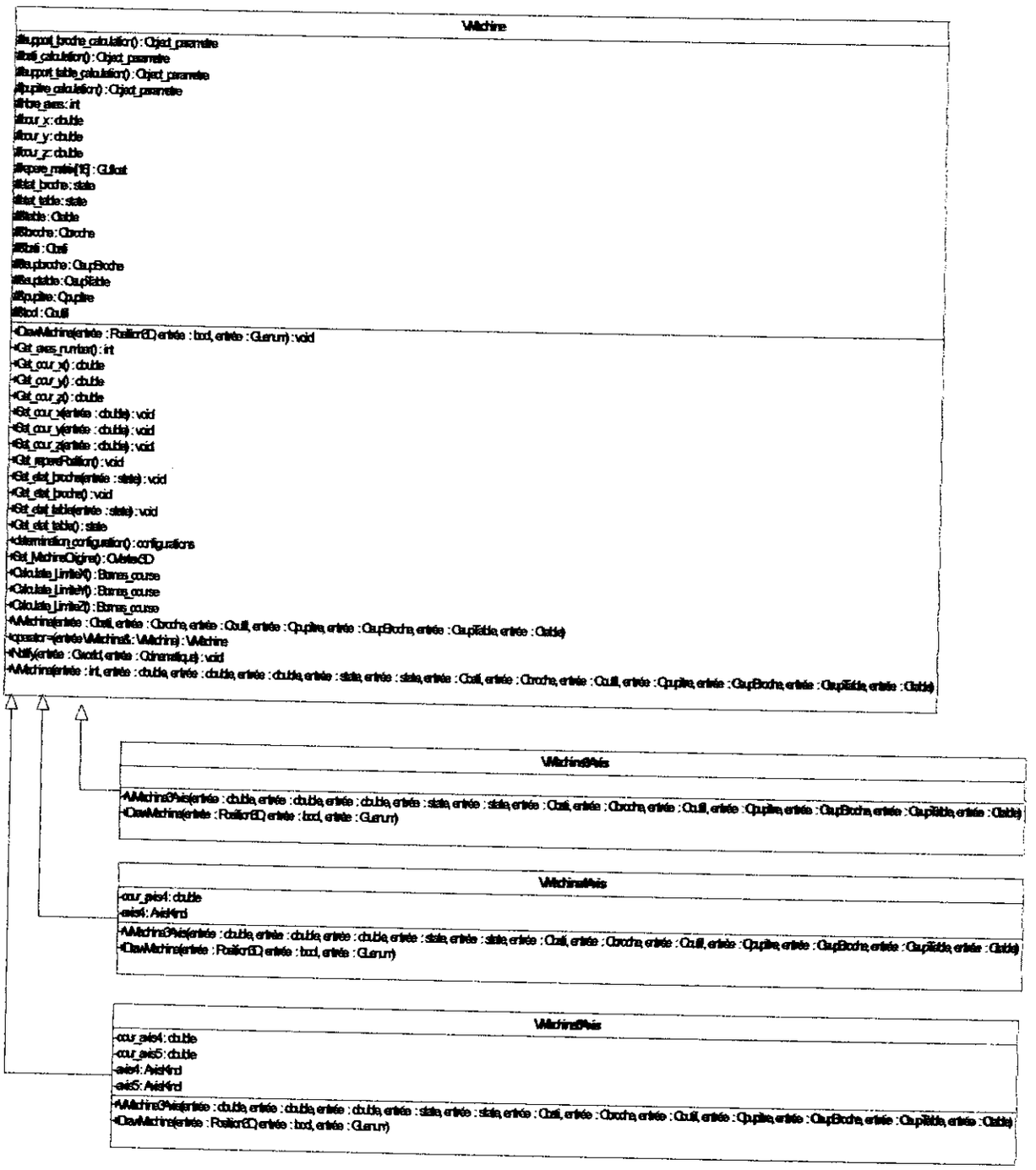


figure IV.16 : *La classe Vmachine et ses sous_classes héritées*

Le diagramme suivant représente la classe Vmachine (Virtual Machine) .
Cette classe permet la construction de la fraiseuse entière grâce à sa méthode DrawMachine() ; et ce par l'envoi de message à ses sous classes sus-citées dans les paragraphes suivantes (Cbati , Cbroche, Ctable...).

Une Vmachine possède les informations suivantes :

- *Nbre_axes* , Le nombre d'axes ;
- *Cour_X* , la course de la table ou de la broche sur l'axe **X**
- *Cour_Y* , la course de la table ou de la broche sur l'axe **Y**
- *Cour_Z* , la course de la table ou de la broche sur l'axe **Z**
- *Etat_broche* , l'état de la broche (fixe, rotation , translation)
- *Etat_table*, l'état de la table (fixe , translation , rotation)
- *Support_broche_calculation* , *support_table calculation*, *bati_calculation* , *pupitre_calculation* ;les dimensions de support broche , de support table , du bâti et du pupitre ,ces dimensions sont calculées en fonction des paramètres de la broche, et de la table .

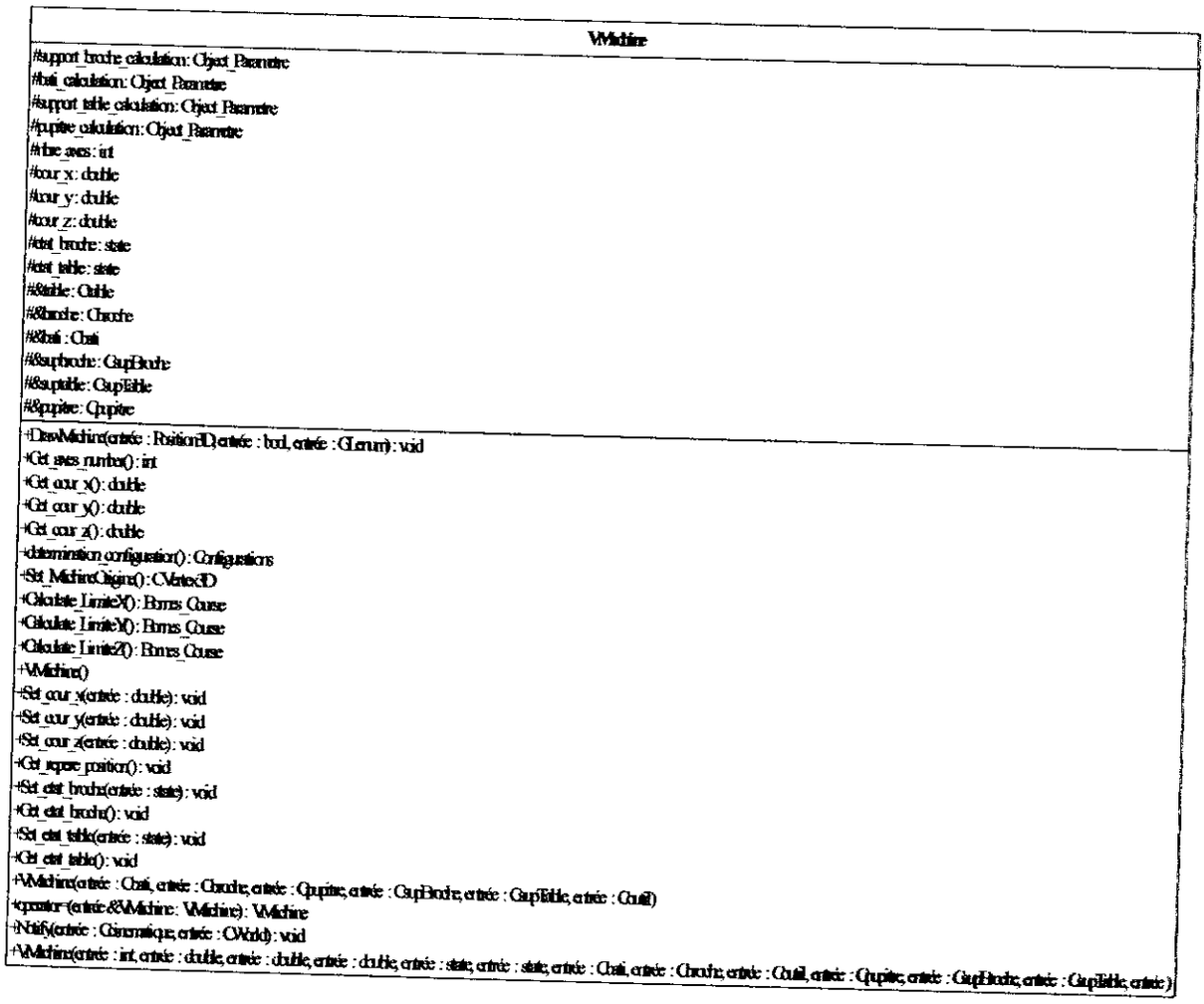


figure IV.17 *: la classe Vmachine*

- ***La classe Cbroche :*** cette classe permet la construction du mandrin de la fraiseuse numérique, par sa méthode DrawBroche ; et ce, par l'envoi de message à la classe Coutil et à l'une ou plusieurs sous classes de la classe Cprimitives.

CBroche
-radius_broche : double -heigh_broche : double -broche_translate : translate -broche_rotate : rotate -cylindre : Ccylinder -disk : Cdisk -box : Cbox -outil : Coutil
+Cbroche(entrée : Coutil) +DrawBroche() : void +Get_axes_translation() : translate +Get_axes_rotation() : rotate +Get_radius_broche() : double +Get_heigh_broche() : double +Cbroche(entrée : GLdouble, entrée : GLdouble, entrée : translate, entrée : rotate, entrée : Coutil, entrée : Coolor) +operator =(entrée : CBroche) : CBroche +Notify() : void +Set_radius_broche(entrée : double) : void +Set_heigh_broche(entrée : double) : void

figure IV.18 *: la classe Cbroche*

- ***La classe Ctable :*** cette classe permet la construction de la table de la fraiseuse numérique, par sa méthode DrawTable ; et ce, par l'envoi de message à la classe Cporte_piece et à l'une ou plusieurs sous classes de la classe Cprimitives.

Ctable
-long_table : double -lar_table : double -haut_table : double -table_translation : translate -table_rotation : rotate -cylindre : Ccylinder -box : Cbox -sphere : Csphere -portepiece : CPorte_piece
+Get_long_table() : double +Get_lar_table() : double +Get_haut_table() : double +DrawTable(entrée : state, entrée : double, entrée : GLenum) : void +Get_axes_translation() : translate +Get_axes_rotation() : rotate +Ctable() +Ctable(entrée : double, entrée : double, entrée : double, entrée : translate, entrée : rotate, entrée : Ccolor) +Set_long_table(entrée : double) : void +Set_lar_table(entrée : double) : void +Set_haut_table(entrée : double) : void +operator =(entrée : Ctable) : Ctable

figure IV.19 *: la classe Ctable*

- ***La classe Cbati :*** cette classe permet la construction du bâti de la fraiseuse numérique , par sa méthode DrawBati ; et ce, par l'envoi de message à l'une ou plusieurs sous classes de la classe Cprimitives

Cbati
-box : Cbox -cylindre : Ccylinder -sphere : Csphere -broche : CBroche -table : Ctable -dfbox : CdfBox -trianbox : Ctrianbox
+Cbati(entrée : Ccolor, entrée : CBroche, entrée : Ctable) +DrawBati(entrée : Object_parametre, entrée : state, entrée : GLenum) : **void** +Cbati(entrée : CBroche, entrée : Ctable) +operator =(entrée : Cbati) : Cbati

figure IV.20 *: la classe Cbati*

- ***La classe Cpupitre :*** cette classe permet la construction du pupitre de la fraiseuse numérique , par sa méthode DrawPupitre ; et ce, par l'envoi de message à l'une ou plusieurs sous classes de la classe Cprimitives

CPupitre
-box : Cbox -cylindre : Ccylinder
+Cpupitre(entrée couleur : Ccolor) +DrawPupitre(entrée : Object_parametre, entrée : Ccolor) : **void** +Cpupitre()

figure IV.21 *: la classe Cbati*

- ***La classe CsupTable :*** cette classe permet la construction du support_table de la fraiseuse numérique , par sa méthode DrawSupTable ; et ce, par l'envoi de message à l'une ou plusieurs sous classes de la classe Cprimitives

CsupTable
-box : Cbox -table : Ctable -cylindre : Ccylinder
+CsupTable(entrée couleur : Ccolor, entrée table : Ctable) +DrawSupTable(entrée : Object_parametre, entrée : state, entrée : double, entrée : GLenum) : void +CsupTable(entrée : Ctable) +operator=(entrée : CsupTable) : CsupTable

***figure IV.22**: la classe CsupTable*

- ***La classe Csupbroche :*** cette classe permet la construction du support_broche de la fraiseuse numérique , par sa méthode DrawSupBroche ; et ce, par l'envoi de message à l'une ou plusieurs sous classes de la classe Cprimitives et à la classe Cbroche.

CsupBroche
-broche : CBroche -box : Cbox -cylindre : Ccylinder -table : Ctable -dfBox : CdfBox
+DrawSupBroche(entrée : Object_parametre, entrée : state, entrée : GLenum, entrée : double) : void +CsupBroche(entrée color : Ccolor, entrée broche : CBroche, entrée : Ctable) +CsupBroche(entrée : CBroche, entrée : Ctable) +operator=(entrée : CsupBroche) : CsupBroche

***figure IV.23**: la classe CsupBroche*

- ***La classe Coutil :*** cette classe permet la construction de l'outil de la fraiseuse numérique , par sa méthode Drawoutil ; et ce par l'envoi de message à l'une ou plusieurs sous-classes de la classe Cprimitives

Coutil
-height_outil : double -radius_outil : double -cylindre : Ccylinder -sphere : Csphere
+Coutil(entrée : double, entrée : double, entrée : Ccolor) +DrawOutil(entrée : GLenum) : void +Get_height_outil() : double +Get_radius_outil() : double +Coutil() +operator =(entrée : Coutil) : Coutil +Set_heigt_outil(entrée : double) : void +Set_radius_outil(entrée : double) : void

***figure IV.24**: la classe Coutil*

- ***La classe CportePiece :*** cette classe permet la construction de la porte_pièce de la fraiseuse numérique , par sa méthode DrawPortePiece ; et ce, par l'envoi de message à l'une ou plusieurs sous classes de la classe Cprimitives.

CPorte_piece
-log_PP : double -larg_PP : double -epaiss_PP : double -box : Cbox -cylindre : Ccylinder -sphere : Csphere -disk : Cdisk
+Cporte_piece(entrée : double, entrée : double, entrée : double, entrée : Ccolor) +DrawPortePiece(entrée : String, entrée : double, entrée : GLenum, entrée : rendu) : void +Get_Log_pp() : double +Get_larg_PP() : double +Get_epaiss_PP() : double

***figure IV.25**: la classe CportePiece*

Le diagramme des classes suivant décrit en détail la VMachine et ses sous-classes :

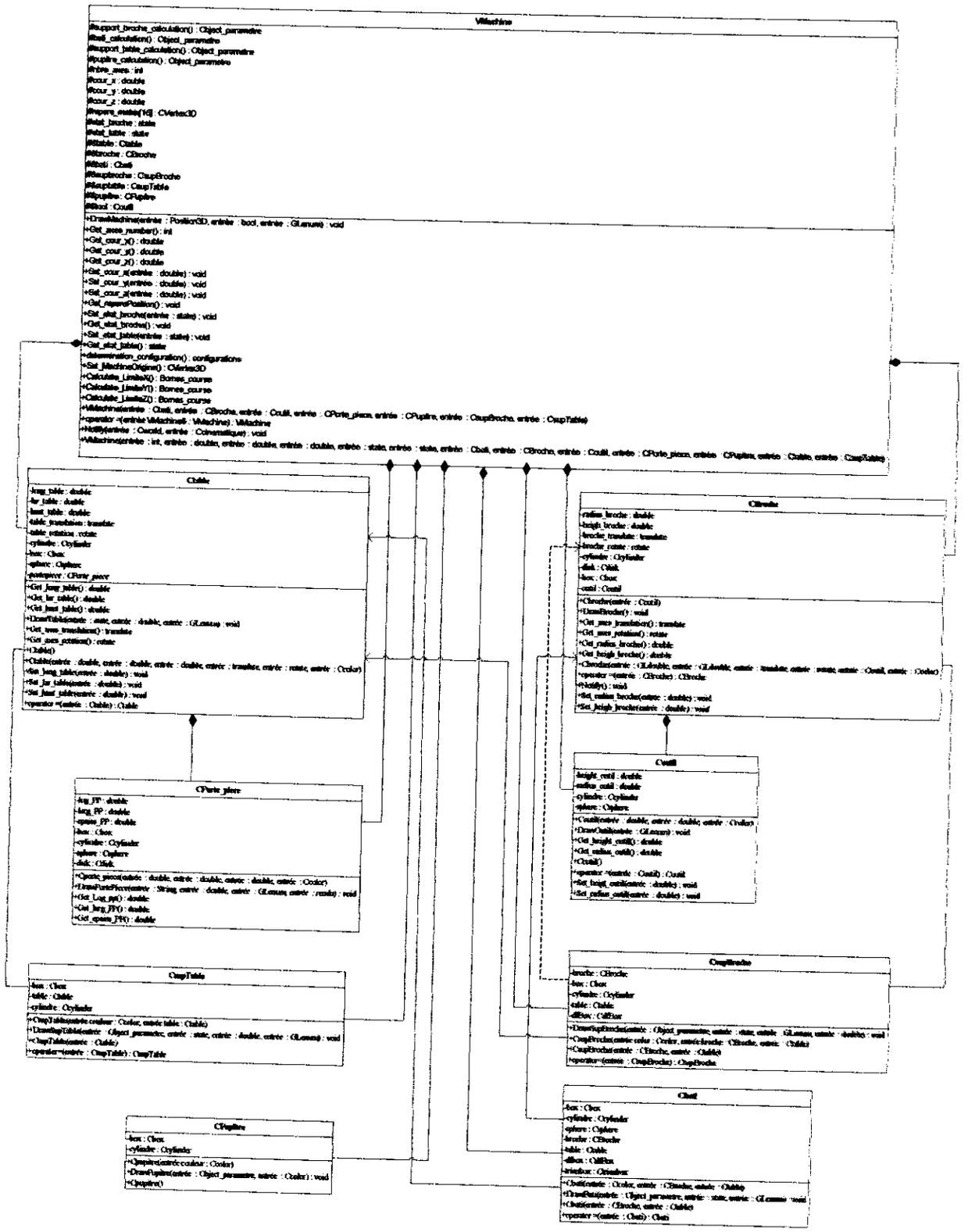


figure IV.26 *: La classe Vmachine et ses sous _classes*

IV.4- Vérification de l'usinabilité d'une surface :

- **La classe Csurfaces_Machine**

Pour étudier l'usinabilité d'une surface gauche sur une fraiseuse numérique , nous avons utilisé la classe Csurfaces_machine , qui fait appel aux deux classes Vmachine3axis et la classe Cpièce.

Cette classe permet la triangulation de la surface avec sa méthode Draw_SurfaceTriangule() , elle permet aussi le calcul de l'aire de surface non visible(1) et non accessible(2) avec les deux méthodes calculate_AireNonVisible() et calculate_AireNonAccessible().

Si l'aire non visible =0 et l'aire non accessible =0
Alors on peut dire que cette surface est usinable sur cette machine
Sinon l'usinage provoque une collision durant l'opération

Et pour étudier la visibilité et l'accessibilité , nous avons besoin d'utiliser la classe Ctriangle .

Csurfaces_machine
-nb_lines : int -nb_Columns : int -nbreS : int -*nuage_surfaceTab : TPoint -*TriangleList : TList
+Csurfaces_machine(entrée nc : int, entrée nl : int, entrée ns : int) +Csurfaces_machine() +Calculate_Radius_Min() : void +SetTriangleList() : void +Draw_SurfaceTriangule() : void +Calculate_AireSurface() : double +Calculate_AireNonVisible() : double +Calculate_AireNonAccessible() : double

figure IV.27 *: Csurfaces_machine*

- **La classe Ctriangle :**

Pour étudier l'usinabilité , on a passé par la triangulation de la surface à usiner ,ce maillage est nécessaire pour faciliter l'étude de la visibilité et l'accessibilité .
La classe Ctriangle décrit une maille, et sa méthode *PointAppartence_Determination(Tpoint)* pour tester l'appartenance d'un point à un triangle, est nécessaire pour étudier la visibilité et l'accessibilité.

Ctriangle
-v1 : Tpoint -v2 : Tpoint -v3 : Tpoint
+Ctriangle(entrée p1 : Tpoint, entrée p2 : Tpoint, entrée p3 : Tpoint) +~Ctriangle() +GetV1() : Tpoint +GetV2() : Tpoint +GetV3() : Tpoint +SetV1(entrée : Tpoint) : void +SetV2(entrée : Tpoint) : void +SetV3(entrée : Tpoint) : void +Calculate_Module(entrée : Tpoint, entrée : Tpoint) : double +Calculate_DroiteParametre(entrée : Tpoint, entrée : Tpoint) : droite_param +Calculate_AireTriangle() : double +PointAppartenance_Determination(entrée : Tpoint) : bool +Calculate_CentreGravite() : Tpoint

figure IV.28 *: la classe Ctriangle*

- **La classe Cpiece**

La classe Cpièce décrit le brut à usiner (la matière entrée avant l'usinage) qui est caractérisé par les trois dimensions : la longueur (brut_long) , la largeur (brut_larg) et la hauteur (brut_haut).

CPiece
-brut_long : double -brut_larg : double -brut_haut : double
+CPiece() +CPiece(entrée lbr : double, entrée larbr : double, entrée hbr : double) +DrawBrut(entrée otab[] : CVertex3D) : void +DrawPiece(entrée rendu) : void

figure IV.29 *: la classe Cpiece*

IV.5- La simulation de l'usinage :

- **La classe SimulateMachining :**

Cette classe, comme son nom l'indique, nous permet de simuler l'usinage , elle possède deux variables de membres privés des classe Ccinematique et Cworld.

SimulateMachining
-*toolPathPoint : PointTrajet* -origines[10] : Cvertex3D -orgprogram : Cvertex3D -*PointList : TStringList -taille_trajet : int -Calculate_depCenter() : Object_parametre #Inverse_Kinematic : Ccenimatique #&scene : Cworld
+SimulateMachining(entrée &InkIni : Ccenimatique) +ReadFile(entrée : AinsiString) : void +Set_toolPathPoint(entrée : PointTrajet*) : void +Set_origines(entrée : Cvertex3D) : void +Get_Origines() : PointTrajet* +Get_toolPathPoint() : PointTrajet* +Get_Origines() : Cvertex3D +SimulateMilling() : void +AjusterPoints() : void +Get_C() : Ccenimatique +Get_S() : Cworld +UpdateS(entrée : Cworld) : void +UpdateC(entrée : Ccenimatique) : void +Notify() : void +operator =(entrée : SimulateMachining) : SimulateMachining

figure IV.30 : *la classe Csimulate Machining*

- **La classe Ccinematique :**

L'étude de cinématique de la machine outil , consiste principalement à traiter les deux organes mobiles : la table et la broche.

Dans cette section on représente la classe *Ccinematique* qui est utilisée pour résoudre la cinématique directe et la cinématique inverse grâce à ses méthodes *DirectKinematicSolver*(), et *inverse_KinematicSolver()*.

Ccenimatique
-degre_liberte : int -longeur_outil : double -&broche : CBroche -&millig_machine : VMachine
+Ccinematique(entrée : CBroche, entrée : VMachine) +Ccenimatique(entrée : int, entrée : double, entrée : CBroche, entrée : VMachine) +Set_degre_libertie(entrée : int) : void +Set_longeur_outil(entrée : double) : void +trasformer_deplacement(entrée : position) : position +Direct_kinimaticSolver(entrée : position) : interpolation +Inverse_kinimaticSolver(entrée : interpolation) : position +operator=(entrée Ccinematique) : Ccinématique +UpdateM(entrée : VMachine) : void +UpdateBR(entrée : CBroche) : VMachine +Notify(entrée : SimulateMachining) : void

figure IV.31 : *la classe Ccinematique*

Le modèle géométrique direct permet de connaître, à partir d'une configuration définie par les coordonnées articulaires, la position du repère de l'effecteur dans le repère de la tache. Lorsqu'on désire commander une structure mécanique quelconque (robot, machine_outil , etc..) le problème se pose dans le sens inverse : on connaît la position que doit atteindre l'effecteur (c'est la tache) : il faut connaître la valeur des coordonnées articulaire qui correspond à cette tache définie. Ce problème conduit à ce que l'on appelle, d'une façon générale, le modèle géométrique inverse ou problème de « l'inversion de coordonnées ».

pour mieux comprendre la cinématique de la machine Nous allons présenter dans la suite un exemple détaillé pour une fraiseuse à cinq axes:

Exemple :

Fraiseuse à cinq axes type RRTTT (ou BAXYZ)

Structure et emploi

La figure VI.5 donne un croquis de ce type de machine dans une architecture dite « à banc en croix » . C'est un type de machine encore assez peu répandu dans l'industrie du fait de son coût élevé ; il est bien adapté à l'usinage de pièces de catégories 1 qui présentent des usinages sous divers angles. Il permet, dans un seul posage , d'accéder à cinq faces de la pièce . Ce type de machine équipe assez systématiquement les cellules ou ateliers automatisés et flexibles de production.

Paramétrage

La structure présente trois axes de translation consécutifs. Nous allons encore adopter un paramétrage spécifique.

La figure VI.6, le résume avec les choix arbitraires effectués. Dans la pratique, la distance d_1 , *dans la liaison (0)/(1)* , est choisie de *valeur nulle.*

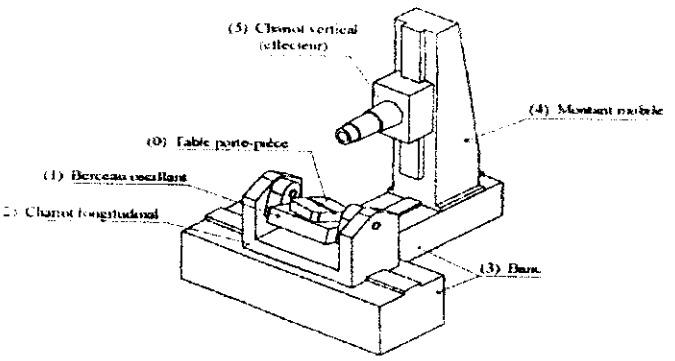


***Figure IV.32** : fraiseuse à cinq axeds de type RRTTT*

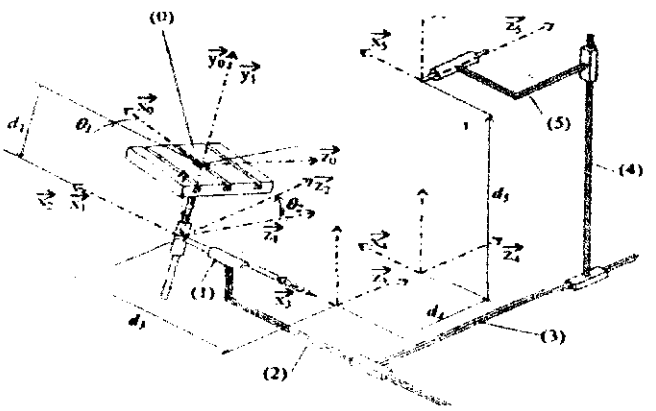


***figure IV.33** :Fraiseuse à cinq axes de type RRTTT (paramétrage)*

Matrices de changement de repère

$$A_1=\begin{pmatrix} c_1 & 0 & s_1 & 0 \\ 0 & 1 & 0 & 0 \\ -s_1 & 0 & c_1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad A_2=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & c_2 & -s_2 & 0 \\ 0 & s_2 & c_2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad A_3=\begin{pmatrix} 1 & 0 & 0 & d_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_4=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad A_5=\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & d_5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (1)$$

Modèle géométrique direct

L'opérateur (R_0/ R_5) est obtenu par le produit des cinq matrices précédentes :

$$(R_0/R_5)= \begin{pmatrix} c_1 & s_1s_2 & s_1c_2 & c_1d_3 + s_1s_2d_5 + s_1c_2d_4 \\ 0 & c_2 & -s_2 & c_2d_5 - s_2d_4 \\ -s_1 & c_1s_2 & c_1c_2 & -s_1d_3 + c_1s_2d_3 + c_1c_2d_4 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (2)$$

Les coordonnées, avec le paramétrage d'Euler, ne s'expriment plus de façon analytique simple. On peut toujours, si nécessaire, les calculer numériquement à partir de l'opérateur.

Modèle géométrique inverse

On se donne cette fois la tache sous forme de l'opérateur (T) :

$$(T)= \begin{pmatrix} s_x & n_x & a_x & p_x \\ s_y & n_y & a_y & p_y \\ s_z & n_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3)$$

Cas d'un effecteur 1D

C'est bien le cas de la machine considérée . La réalisation de la tache se résume à l'équation :

$$\begin{pmatrix} s_1c_2 & c_1d_3 + s_1s_2d_5 + s_1c_2d_4 \\ -s_2 & c_2d_5 - s_2d_4 \\ c_1c_2 & -s_1d_3 + c_1s_2d_5 + c_1c_2d_4 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} a_x & p_x \\ a_y & p_y \\ a_z & p_z \\ 0 & 1 \end{pmatrix} \quad (4)$$

L'identité des termes (2,1) donne $\sin\theta_2 = -a_y$ d'ou les deux solution en θ_2 :

$$\theta_2 = Arcsin(-a_y)$$
$$\theta'_2 = \pi - Arcsin(-a_y) \qquad (5)$$

La solution en θ_1 est alors unique :

$$\theta_1 = ATAN\,2(a_x / \cos\theta_2 , a_z / \cos\theta_2) \qquad (6)$$

- *L'accessibilité de l'outil aux surfaces usinées impose que l'axe $\overrightarrow{Z_5}$ ait une composante positive sur $\overrightarrow{Y_0}$. la course angulaire de l'axe (2) est en fait limitée , dans la pratique , à un intervalle $-\pi/2 \le \theta_2 \le 0$. de ce fait on peut ne retenir qu'une seule configuration : d'ou la seule solution retenue : $\theta_2 = Arcsin(-a_y)$*

Le modèle inverse se résume alors , pour l'effecteur 1D , à la solution unique :

$$\theta_1 = ATAN\,2(a_x / c_2 , a_z / c_2)$$
$$\theta_2 = Arcsin(-a_y)$$
$$d_3 = c_1 p_x - s_1 p_z$$
$$d_4 = s_1 c_2 p_x + c_1 c_2 p_z - s_2 p_y \qquad (7)$$
$$d_5 = s_1 s_2 p_x + c_1 c_2 p_z + c_2 p_y$$

Modèle cinématique

La tache a été définie par la matrice de l'expression (3). La variation de la tache est normalement exprimée par les différentielles des termes de cette matrice ou leurs dérivées par rapport au temps . Le modèle cinématique correspondant peut être défini par dérivation à partir des expressions ci_dessus. Explicitons le modèle inverse

$$\begin{pmatrix} \theta_1 \\ \theta_2 \\ d_3 \\ d_4 \\ d_5 \end{pmatrix} = \begin{pmatrix} J_{11} & 0 & J_{13} & 0 & 0 & 0 \\ 0 & J_{22} & 0 & 0 & 0 & 0 \\ J_{31} & 0 & J_{33} & J_{34} & 0 & J_{36} \\ J_{41} & J_{42} & J_{43} & J_{44} & J_{45} & J_{46} \\ J_{51} & J_{52} & J_{53} & J_{54} & J_{55} & J_{56} \end{pmatrix}$$

En posant : $R = (1 - a_y^2)^{1/2}$ et $C = a_x^2 - a_z^2$, l'expression des termes est :

$J_{11} = a_z / C$

$J_{13} = - a_x / C$

$J_{22} = -1 / R$

$J_{31} = -a_z (s_1 p_x + c_1 p_z) / C$

$J_{33} = a_x (s_1 p_x + c_1 p_z) / C$

$J_{34} = c_1$

$J_{36} = -s_1$

$J_{41} = c_2 a_z (c_1 p_x - s_1 p_z) / C$

$J_{42} = s_2 (s_1 p_x + c_1 p_z) / R + c_2 p_y / R$

$J_{43} = -c_2 a_x (c_1 p_x - s_1 p_z) / C$

$J_{44} = s_1 c_2$

$J_{45} = -s_2$

$J_{46} = c_1 c_2$

$J_{51} = c_2 a_z (c_1 p_x - s_1 p_z) / C$

$J_{52} = s_2 (s_1 p_x + c_1 p_z) / R + c_2 p_y / R$

$J_{53} = -c_2 a_x (c_1 p_x - s_1 p_z) / C$

$J_{54} = s_1 c_2$

$J_{55} = -s_2$

$J_{56} = c_1 c_2$

A l'aide de cette expression il sera possible d'apporter une modification à la position de l'effecteur en programmant les variations articulaires correspondantes.

- **La classe Ccamera :**

Le diagramme ci-dessous représente la classe générale *Ccamera* qui implémente l'interface *Camera_Setting*. Cette camera permet d'appliquer des translations et des rotations autour les axes X,Y,Z des objets de la scène que ce soit la machine ou la pièce .

Ccamera
-m[16] : GLfloat -localm[16] : GLfloat -movespeed : int -rollspeed : int -roll : int
+Ccamera() +moveLeft() : void +moveRight() : void +moveUp() : void +moveDown() : void +moveForward() : void +moveBack() : void +pitchUp() : void +pitchDown() : void +yawRight() : void +yawLeft() : void +rollLeft() : void +rollRight() : void +CameraReset() : void +SetCameraPosition(entrée : int, entrée : int, entrée : int) : void +SetMovementSpeed(entrée : int) : void +SetRollSpeed(entrée : int) +MachinePitchUp() : void +MachinePitchDown() : void +MachineYawRight() : void +MachineYawLeft() : void +MachineRollLeft() : void +MachineRollRight() : void +world_to_local() : void +local_to_world() : void +local() : void +world() : void +ModelCoordinates() : void +WorldCoordinates() : void +inverse_roll() : void +get_matrix() : void

figure IV.34 *:la classe Ccamera*

V- LA VUE DYNAMIQUE DU SYSTEME :

Les diagrammes des séquences décrivent les interactions entre objets . Normalement , un diagramme de séquence sert à décrire un déroulement des évènement d'un cas d'utilisation.

V.1- La création de la machine :

Le diagramme de séquence suivant (IV.35)décrit le déroulement du cas d'utilisation de la création de la machine,

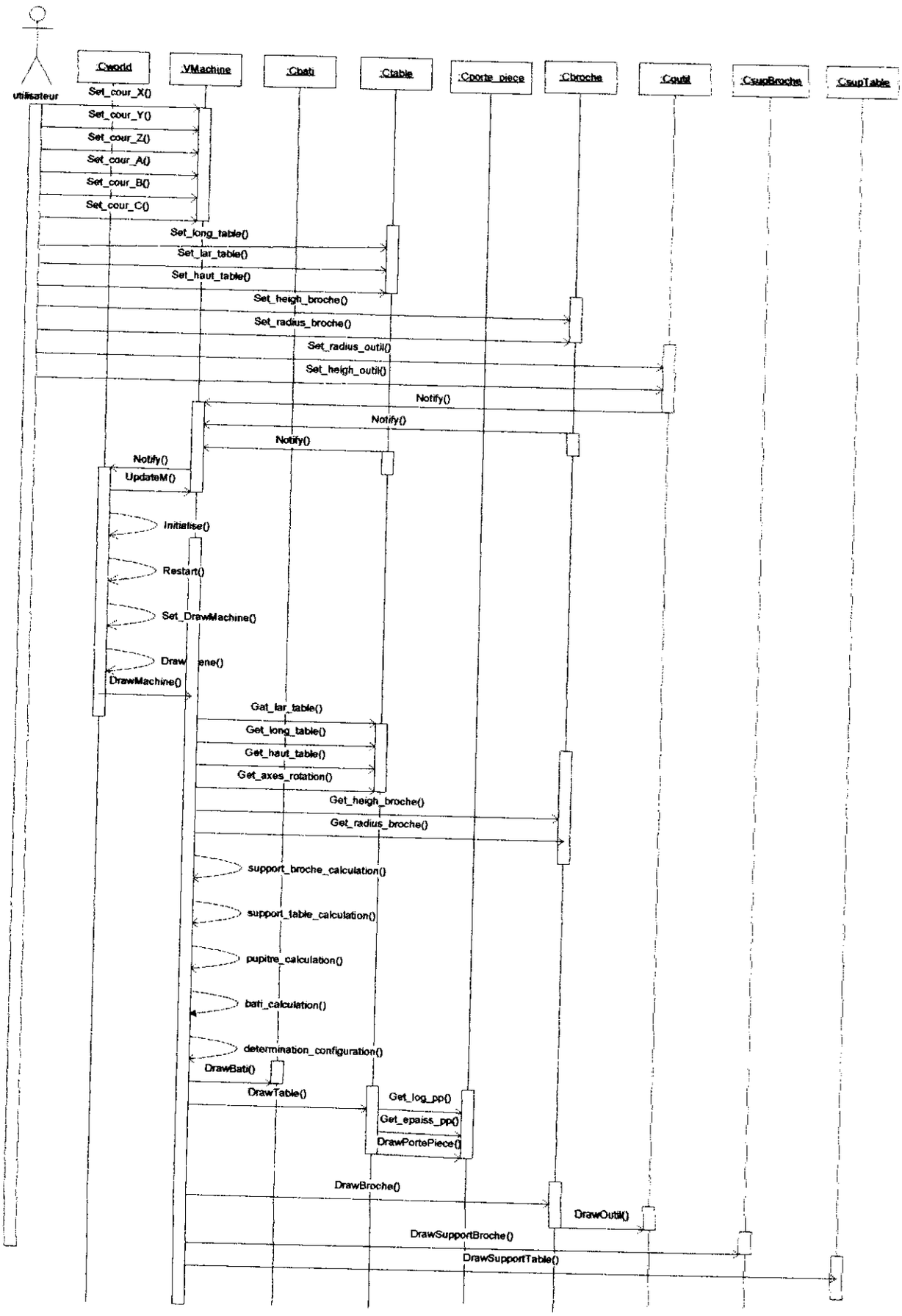


***Figure IV.35** :le diagramme de séquence de la création de la machine*

- ***La création de la porte pièce :***

  - ***Cas d'une table normale :***

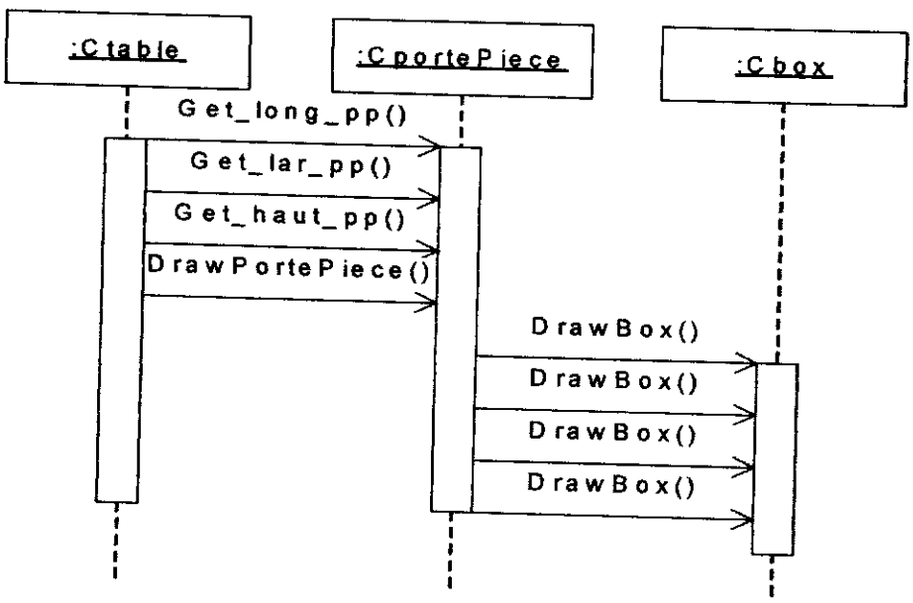


Figure IV.36 *:le diagramme de séquence de la création du porte_pièce*

  - ***Cas d'une table tournante :***

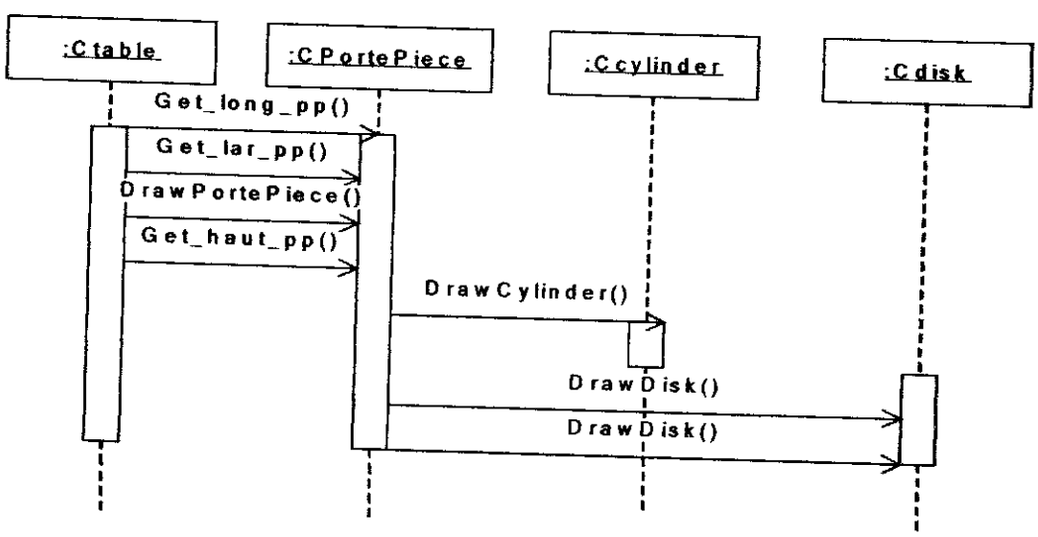


Figure IV.37 *:le diagramme de séquence de la création du porte_pièce*

- ***La création de la table :***

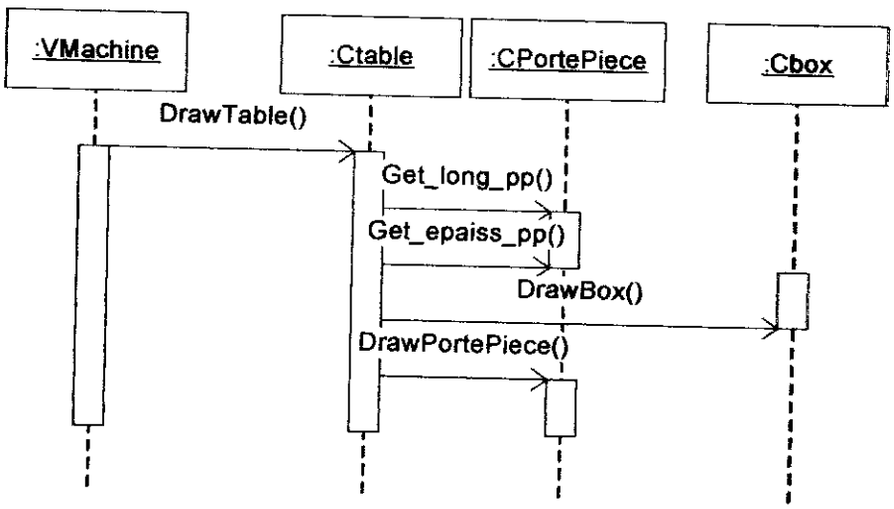


figure IV.37 : *le diagramme de séquence de la création d'une table normale*

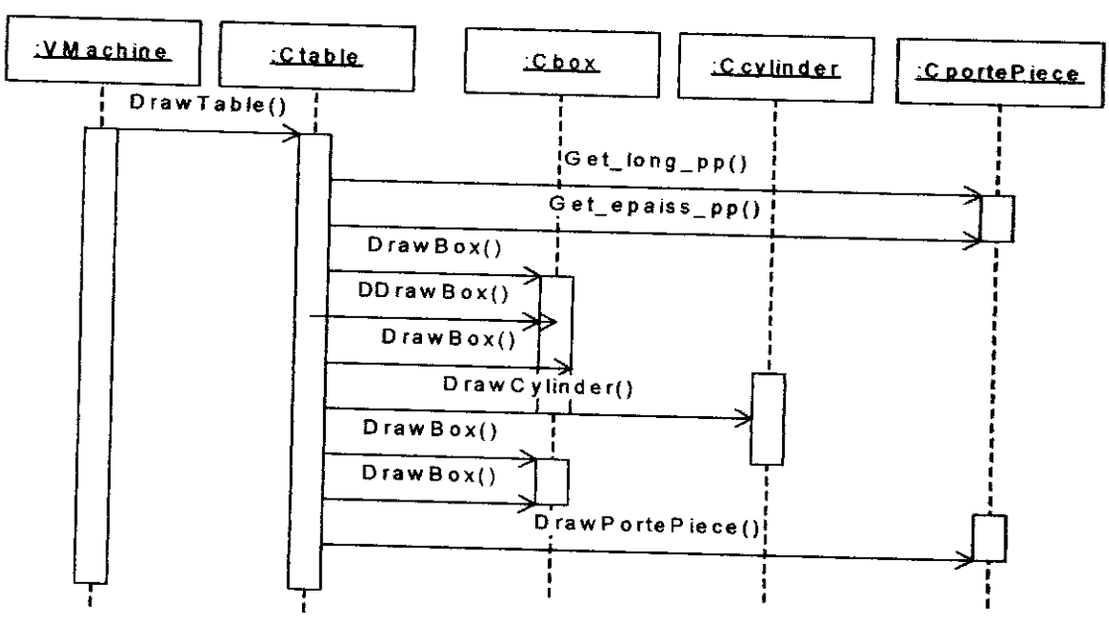


figure IV.38 : *le diagramme de séquence de la création d'une table tournante*

- ***La création du pupitre :***

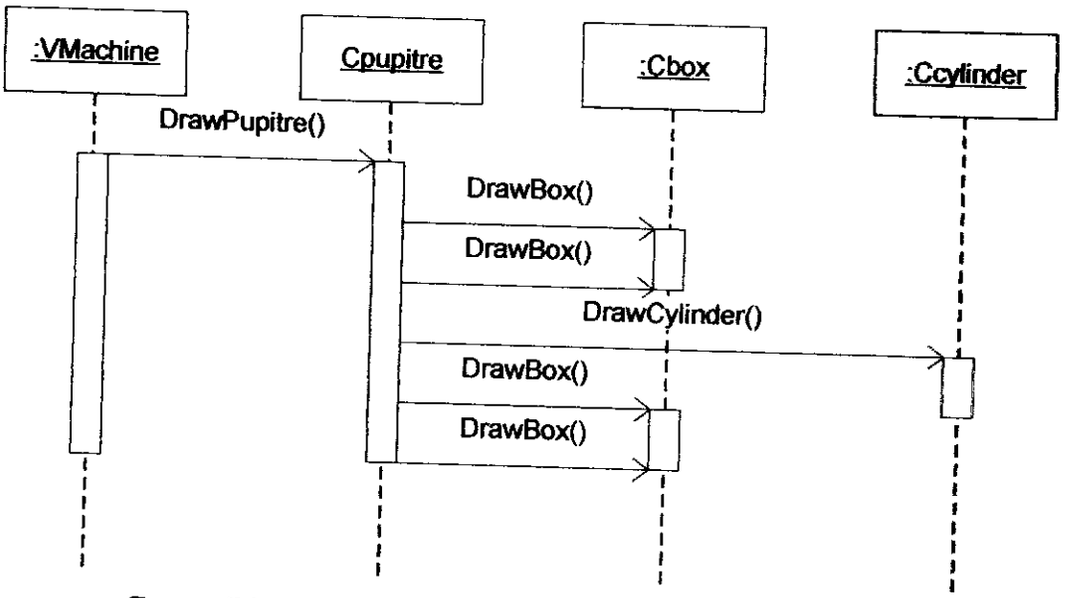


***figure IV.39** : le diagramme de séquence de la création du pupitre*

- ***La création de l'outil :***

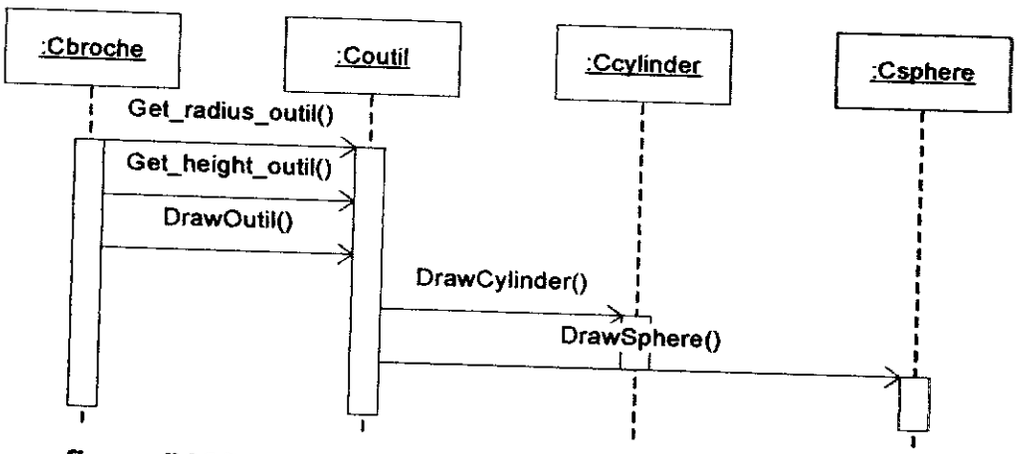


***figure IV.39** : le diagramme de séquence de la création de l'outil*

- ***La création du Broche :***

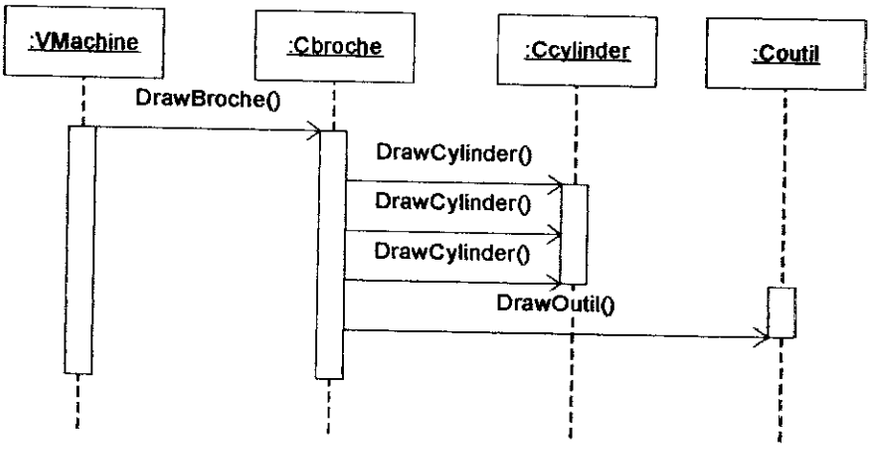


figure IV.40 : *le diagramme de séquence de la création du broche*

- ***La création du support_broche :***

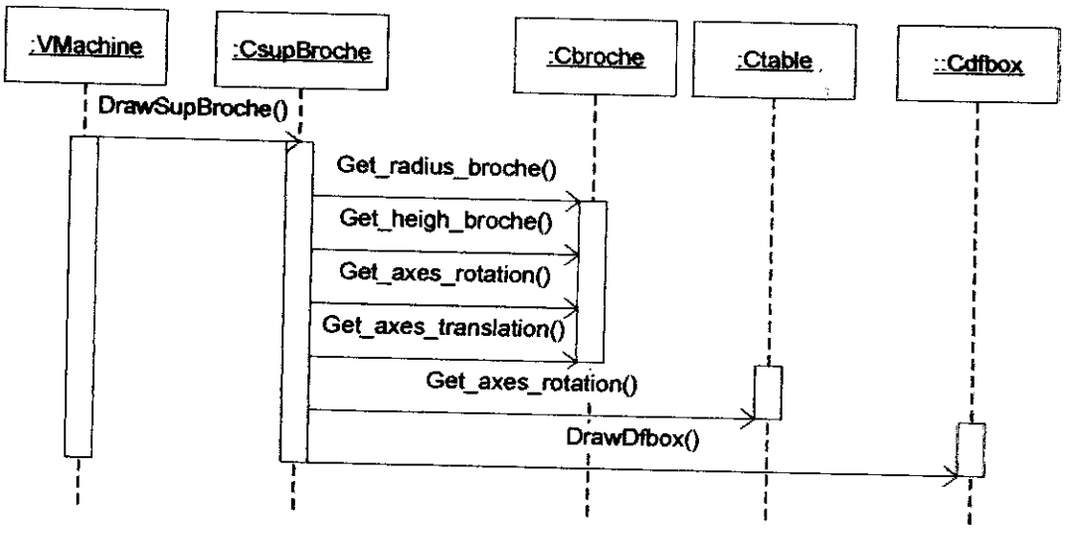


figure IV.41 : *le diagramme de séquence de la création du support_broche*

- ***La création du support_table :***

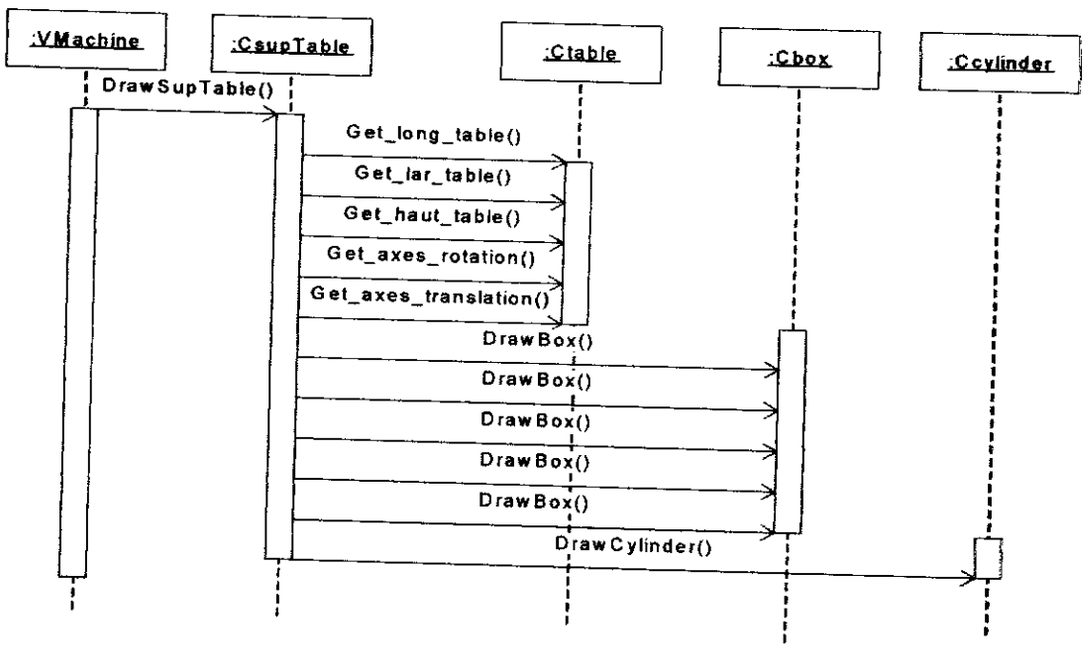


figure IV.42 : *le diagramme de séquence de la création du support table*

- ***La création du bâti :***

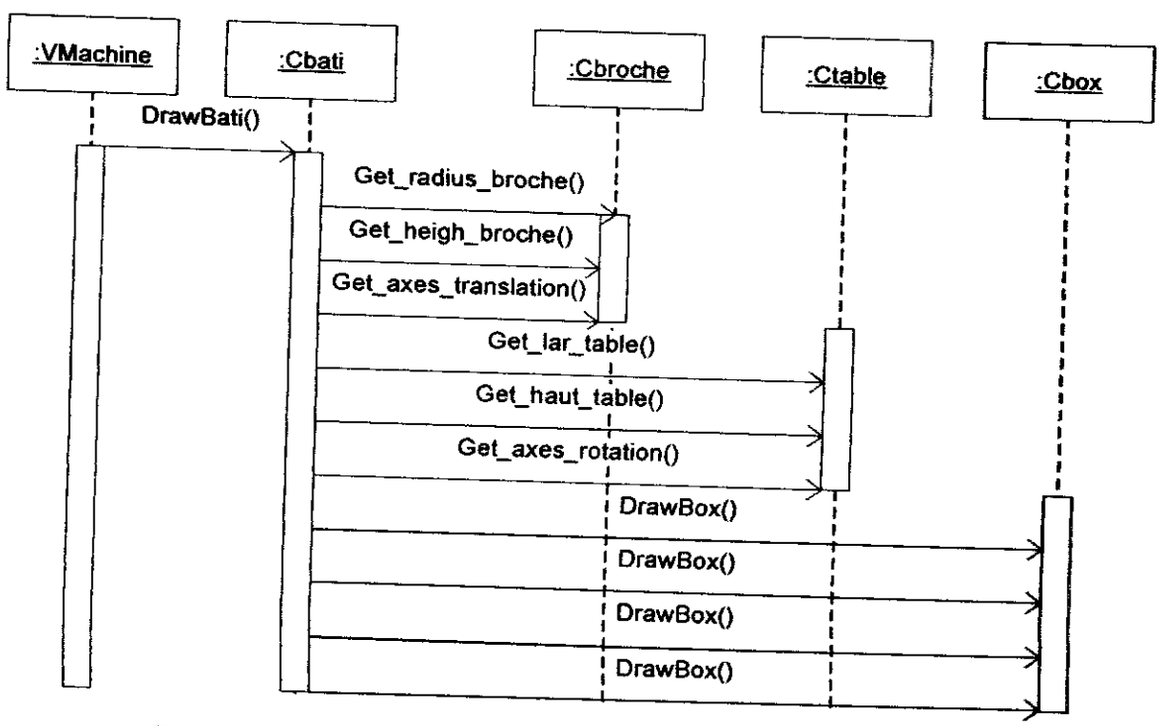


figure IV.43 : *le diagramme de séquence de la création du bâti*

V.2- Le contrôle de la machine :

Le control de la machine dans notre système se fait manuellement en résolvant la cinématique directe ou bien automatiquement en résolvant la cinématique inverse

Le diagramme séquence (figure IV.44) suivant décrit le déroulement des cas d'utilisation de control manuel de la machine .

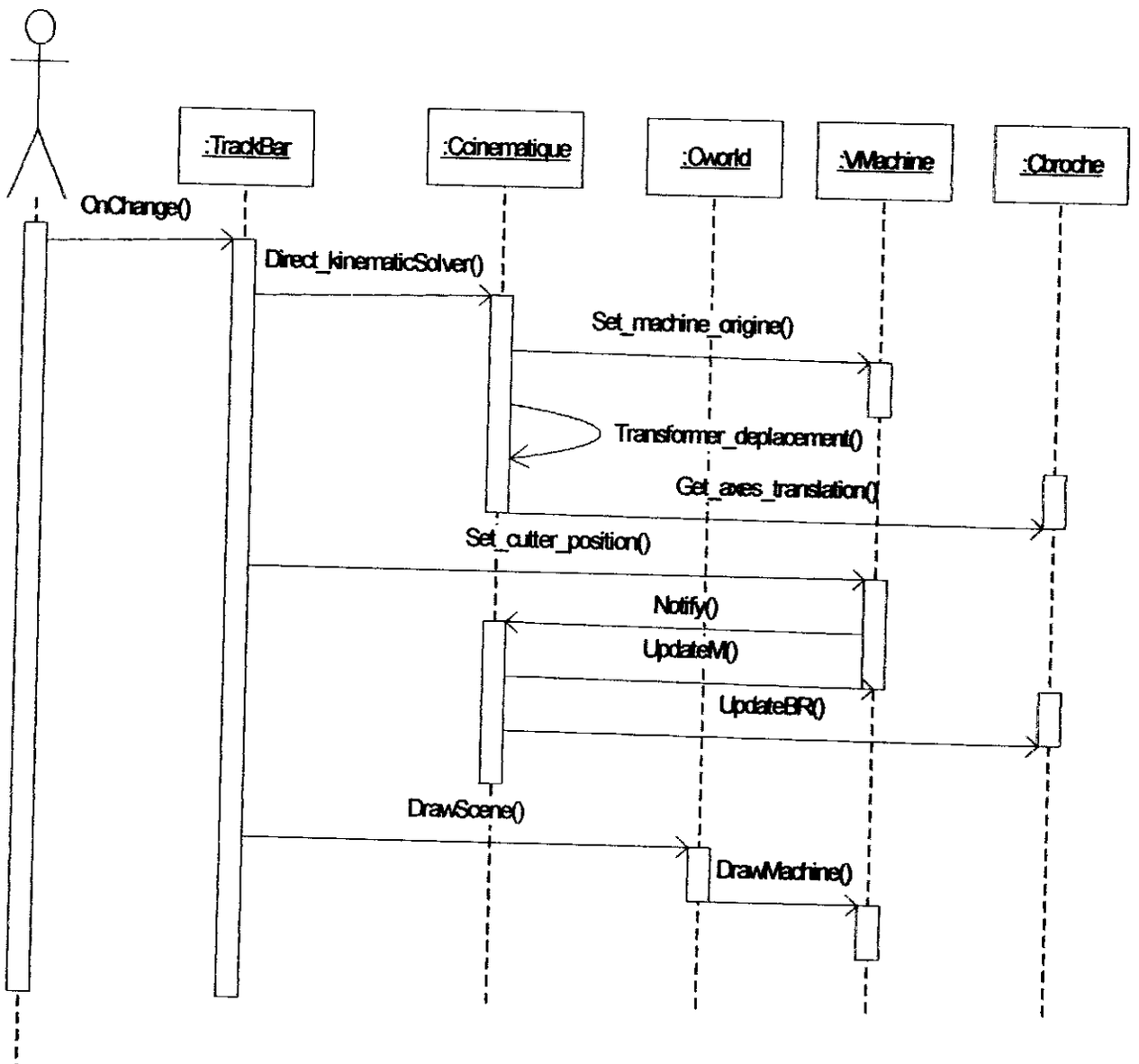


figure IV.44 : *le diagramme de séquence du contrôle manuel de la machine*

Le diagramme séquence (figure IV.45) suivant décrit le déroulement des cas d'utilisation de control automatique de la machine .

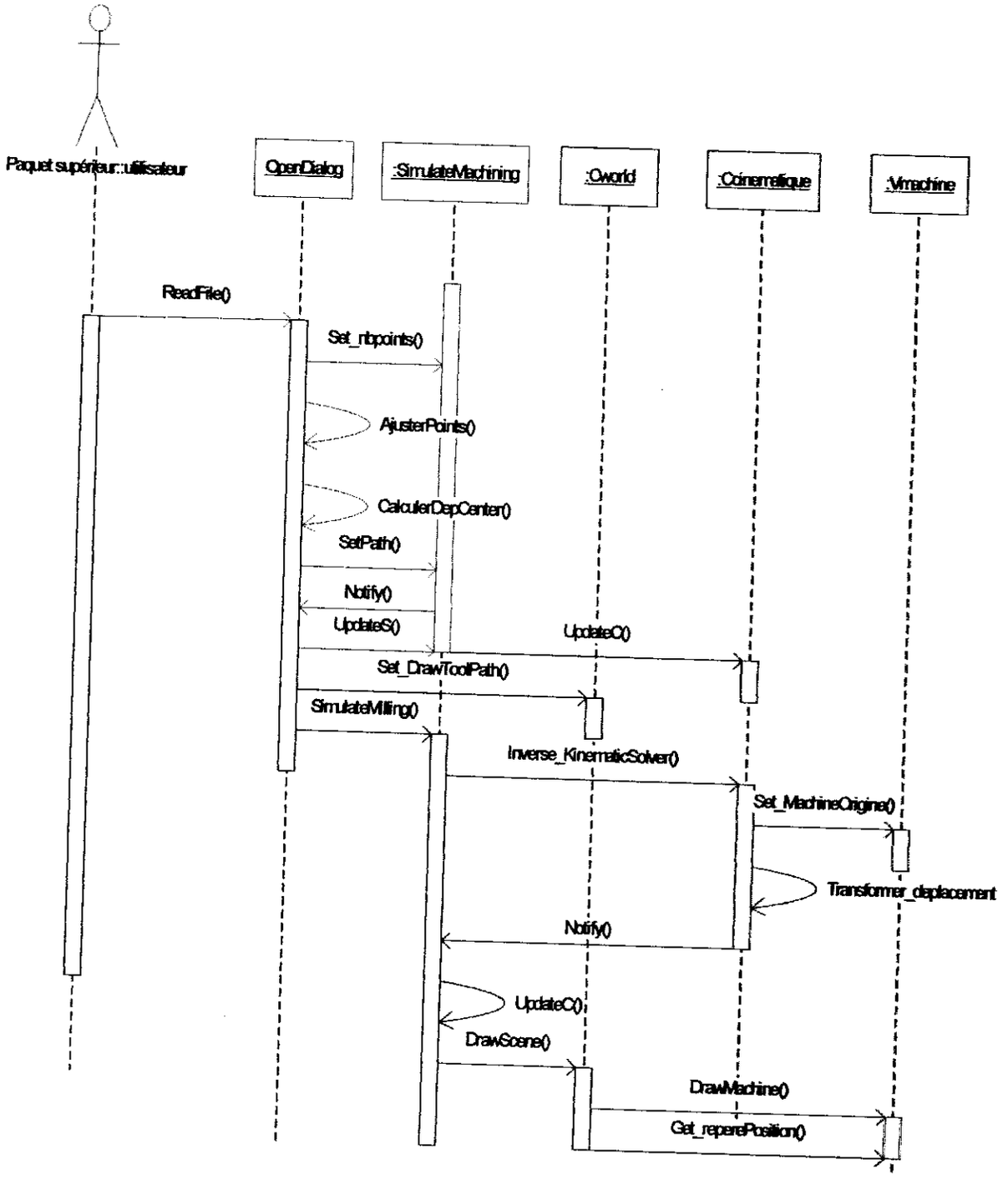
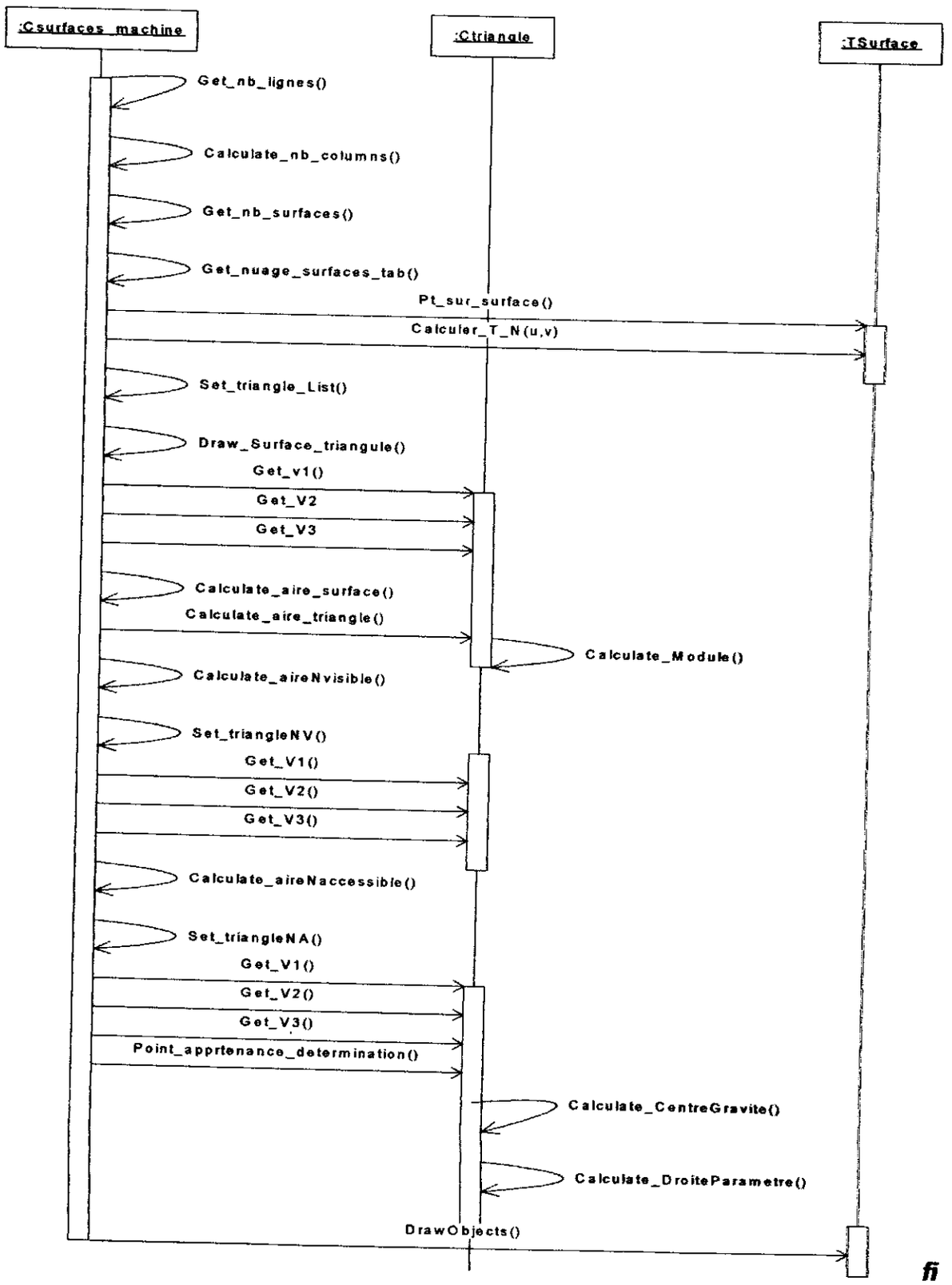


figure IV.45 : *le diagramme de séquence du contrôle automatique de la machine*

V.3- La vérification de l'usinabilité :

Le diagramme de séquence suivant décrit le déroulement des cas d'utilisation du module de la vérification de l'usinabilité



fi

gure IV.46 : *le diagramme de séquence de la vérification de l'usinabilité des surfaces*

V.4- La simulation de l'usinage :

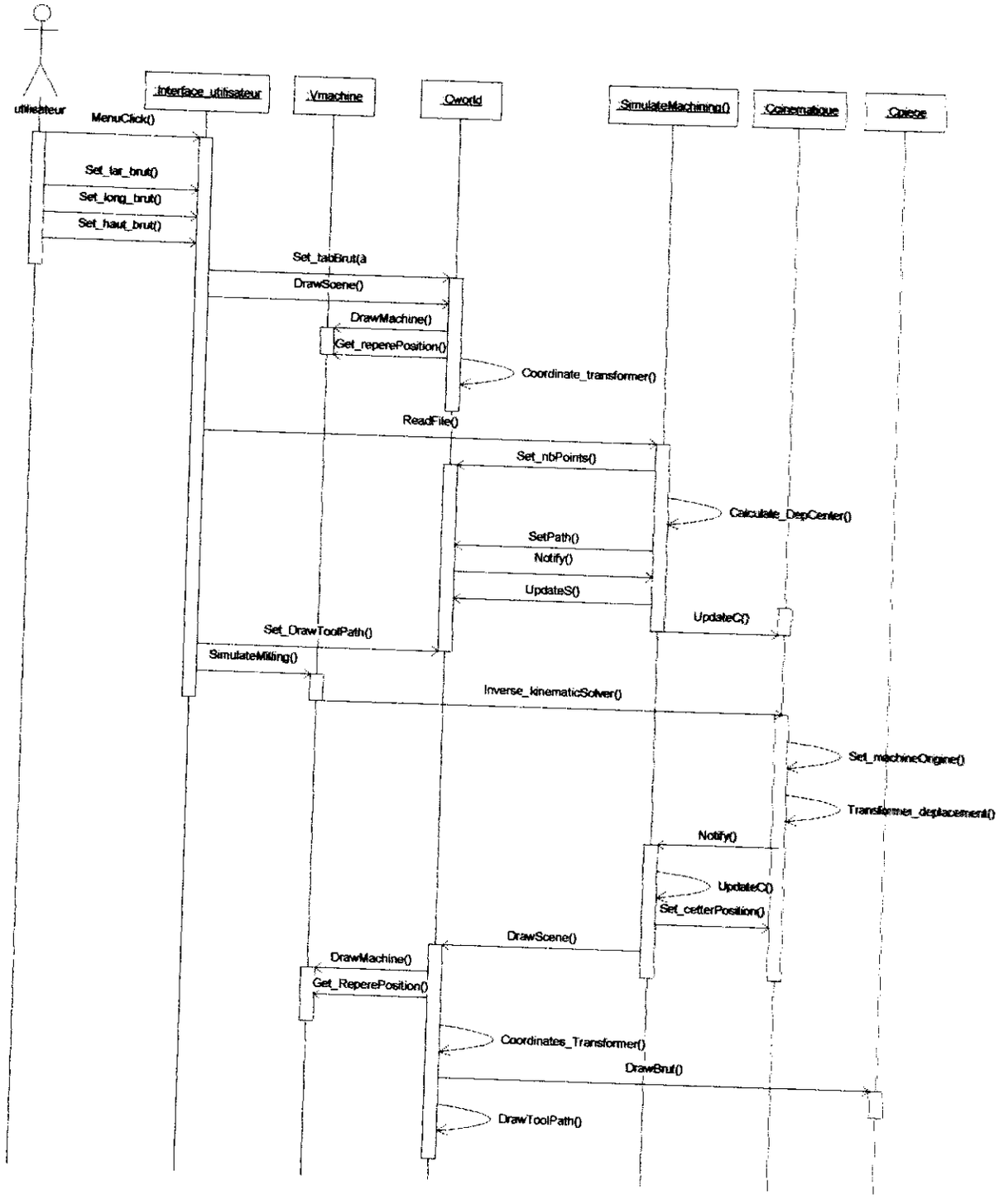


figure IV.47 : *le diagramme de séquence de la simulation de l'usinage*

V- CONCLUSION :

En guise de conclusion de cette partie, nous pouvons dire qu'on a décrit tous les objets (classes et sous classes) nécessaires pour modéliser notre système en utilisant le diagrammes des classes du langage UML, et les différentes interactions qui existent entre ces classe en utilisant le diagramme des séquences .

Les tests

5

DANS CE CHAPITRE

- TESTS INTERMIDIAIRES

I- INTRODUCTION

La démarche extrêmement itérative proposée par XP[1] utilise intensivement les tests unitaires. Nous avons écrit ces tests en même temps que le code lui-même, pour spécifier et valider le comportement de chaque portion de code ajoutée. Quasiment chaque classe de l'application possède une classe jumelle de test.

Dans cette partie de notre mémoire, nous allons présenter un ensemble de résultats obtenus lors du développement de notre système.

II- TESTS INTERMEDIAIRES :

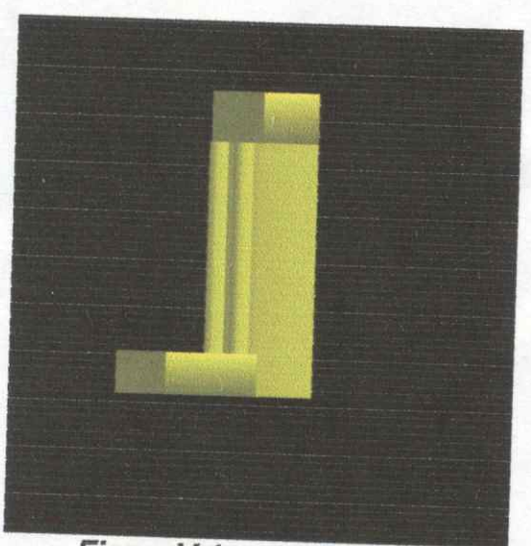
II.1- La création des machines :

La première étape de notre application consistait à concevoir des formes simples 3D.

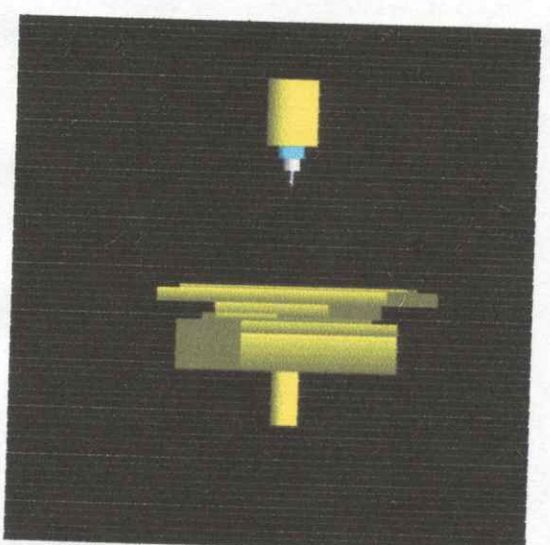
Après la construction des formes de base, nous sommes arrivées à construire l'ensemble des organes d'une machine, et les visualiser :

- en Fil de fer (en lignes).
- en point.
- Ombrée (solide)

Les figures suivantes(figures V.1,V.2,V.3,V.4) sont des captures d'écran montrent des organes des fraiseuses numériques.



***FigureV.1** :un bâti*



***FigureV.2** : une broche et un support_table*

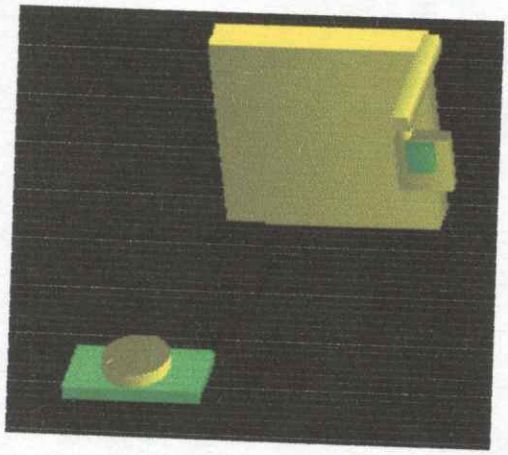


Figure V.3 *:un pupitre et une table tournante*

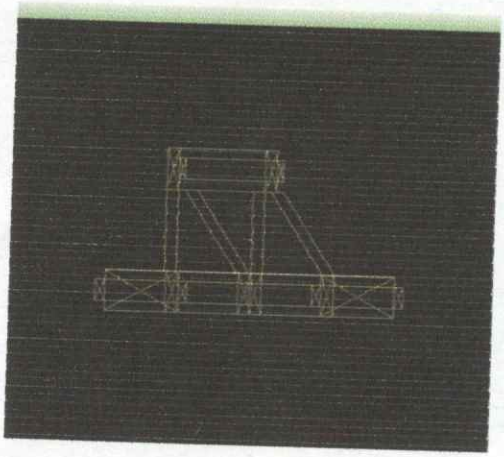


Figure V.4 *: un bâti*

Avec la construction des différents organes de la machine nous sommes arrivées à construire la machine entière.

Pour construire une machine , il faut entrer les paramètres de la machine , de la table, de la broche et de l'outil et donner les axes de déplacements pour les organes mobiles.

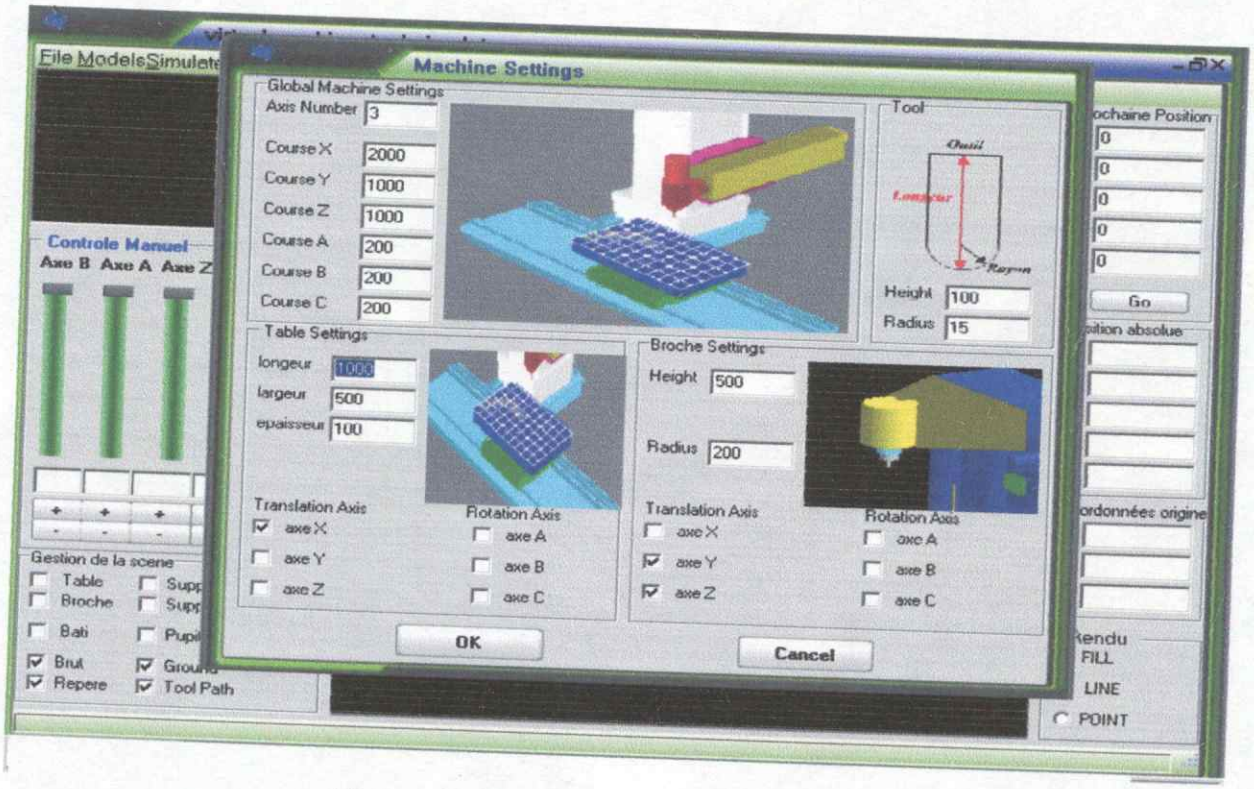


Figure 5 *:les paramètres de la machine*

Dans la suite nous allons présenter quelques types des fraiseuses conçues par notre système

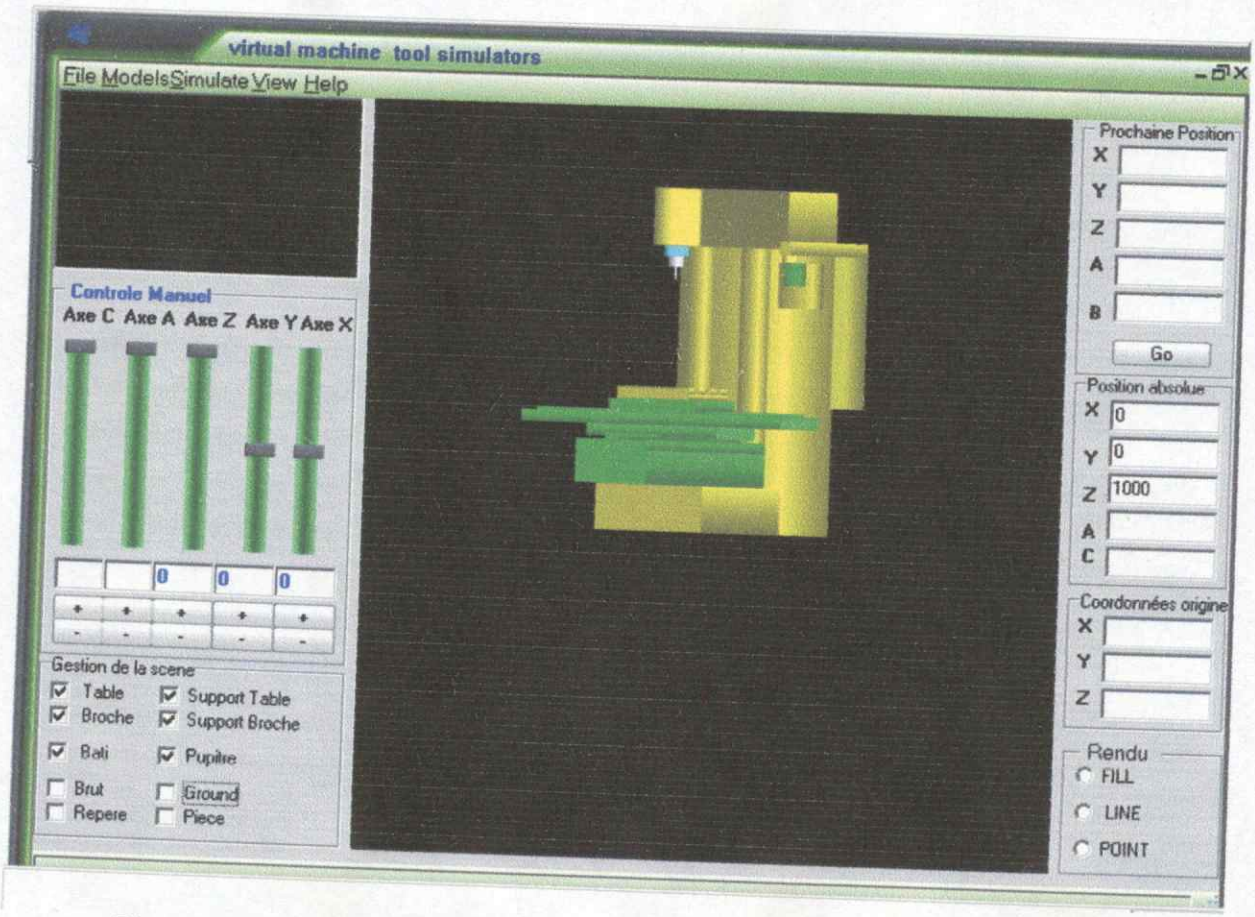


Figure V.6 *:Capture d'écran montre une fraiseuse à trois axes ombrée*

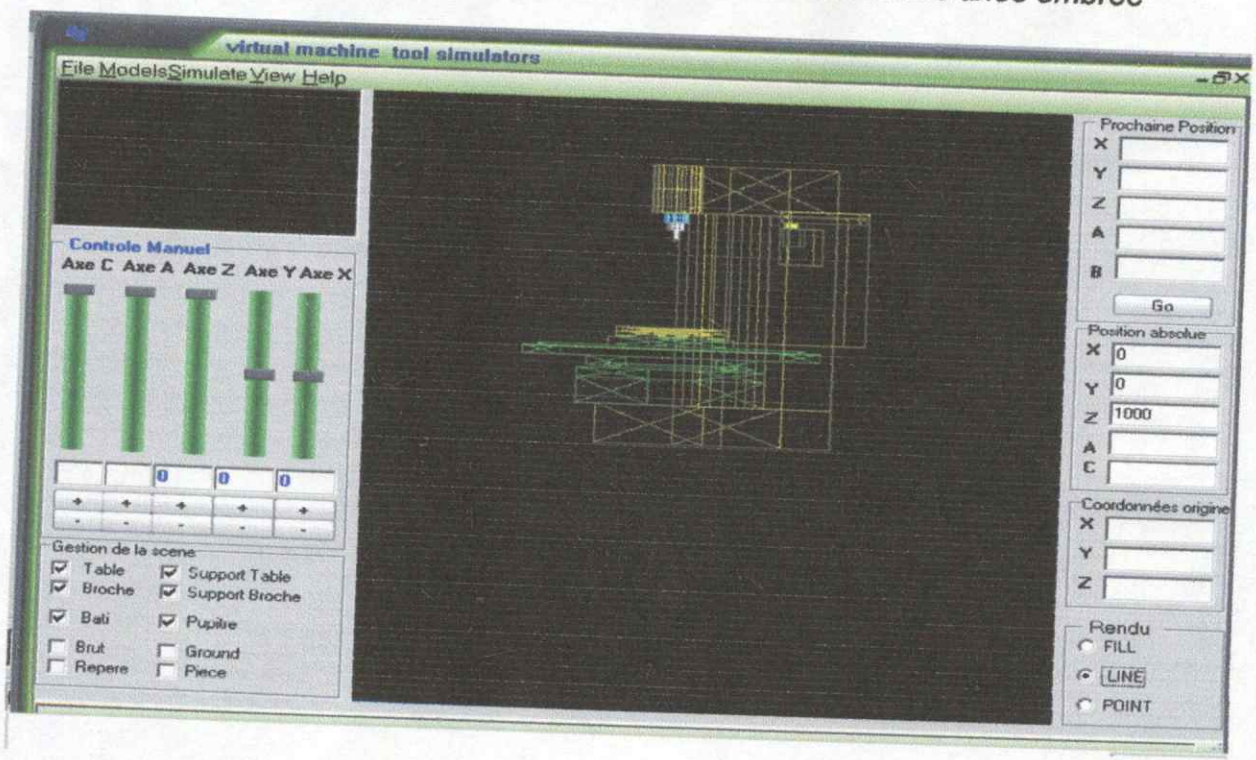


Figure V.7 : *Capture d'écran montre une fraiseuse à trois axes en lignes*

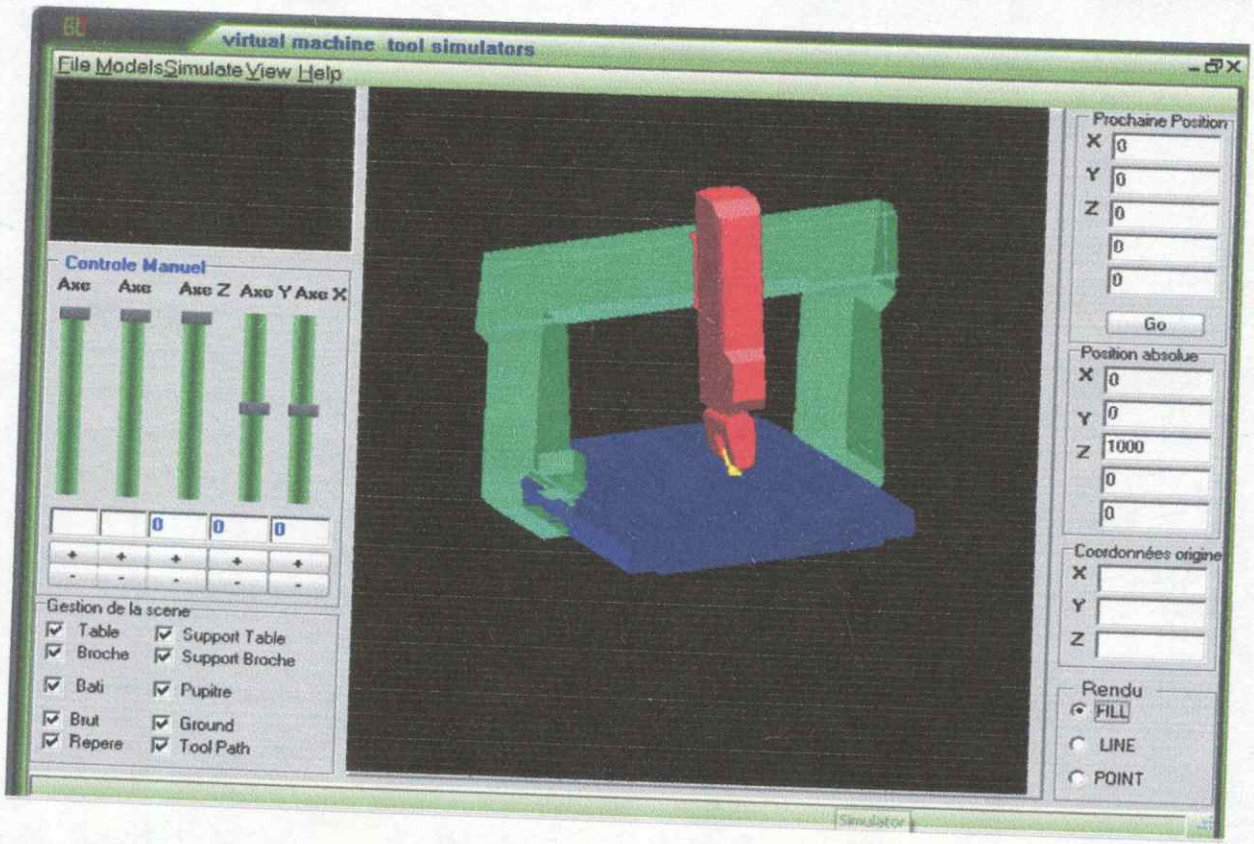


***Figure V.8** : Capture d'écran montre une fraiseuse à cinq axes ombrée*

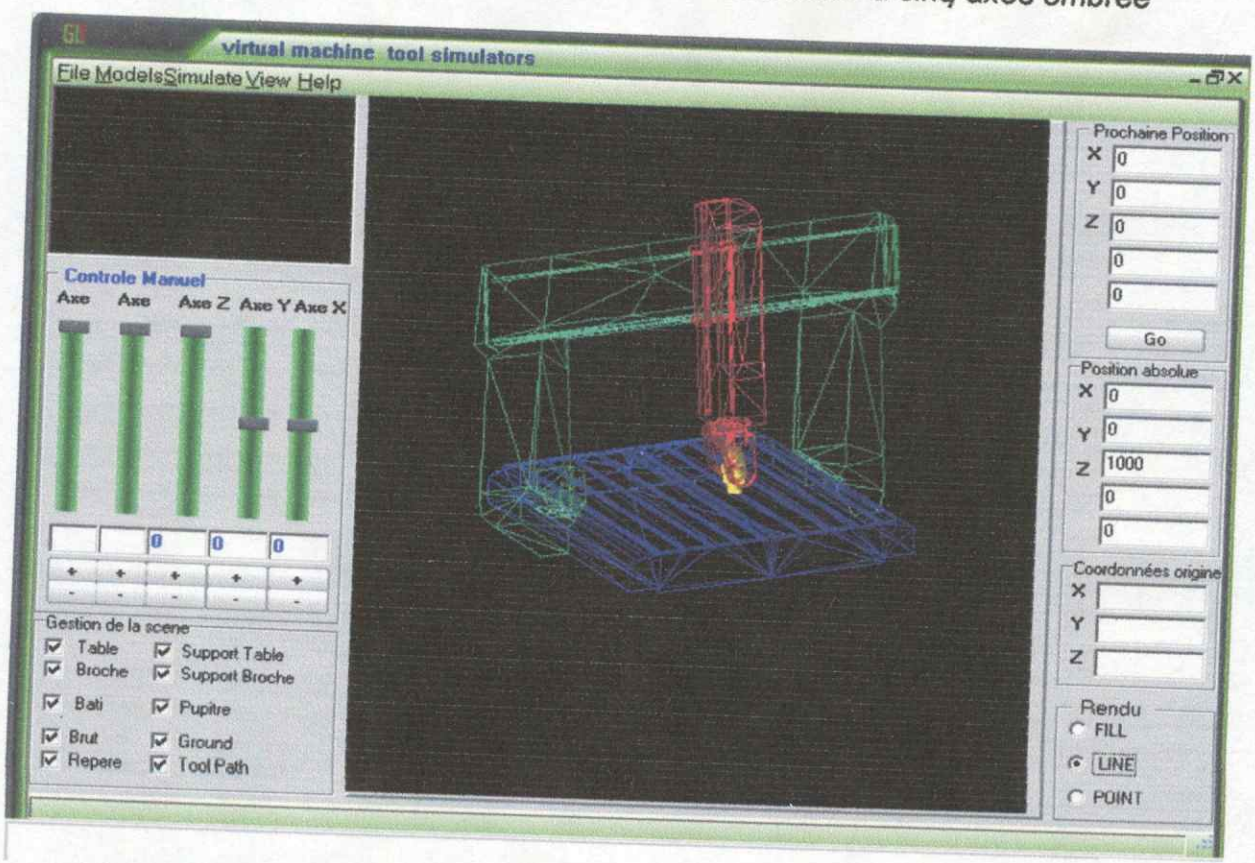


***Figure V.9** : Capture d'écran présente une fraiseuse à cinq axes en lignes*

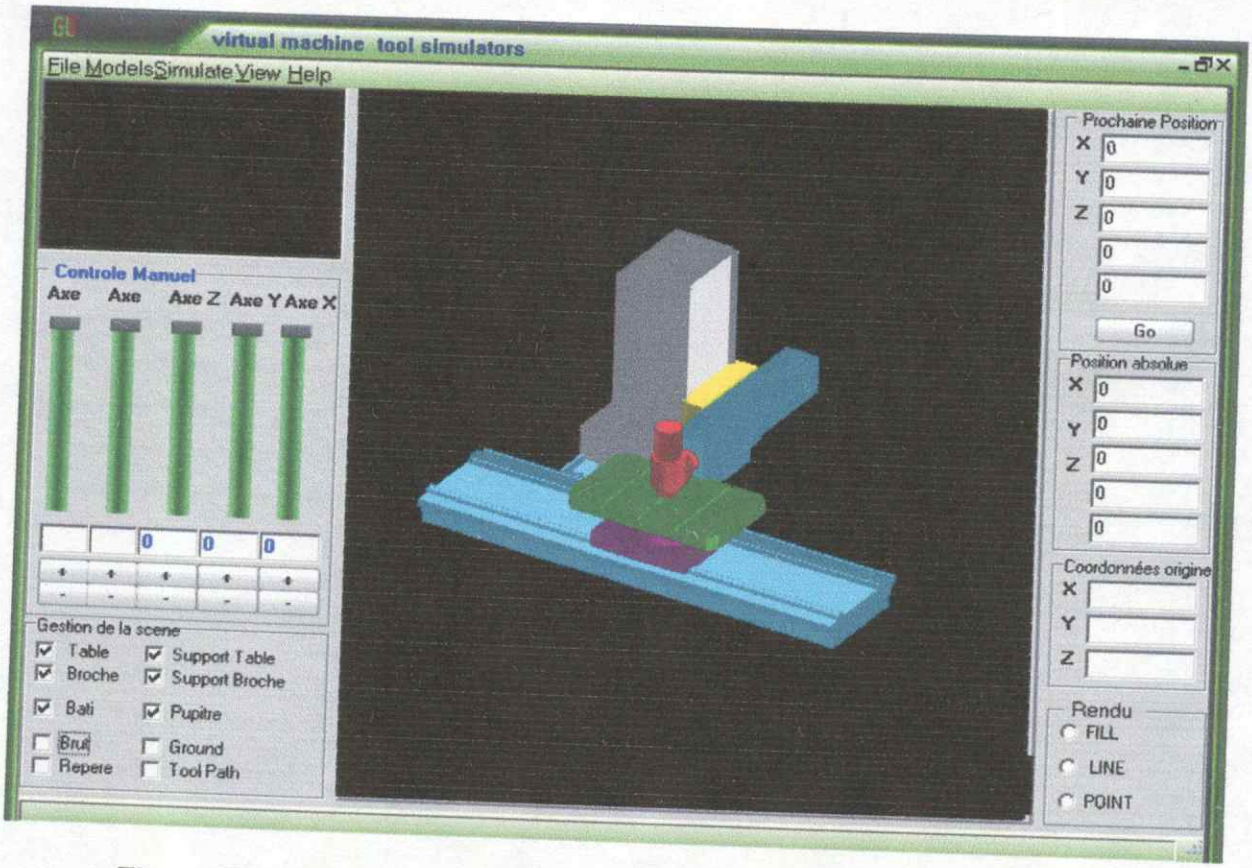


Figure V.10 : *Capture d'écran montre fraiseuse à quatre axes ombrée*

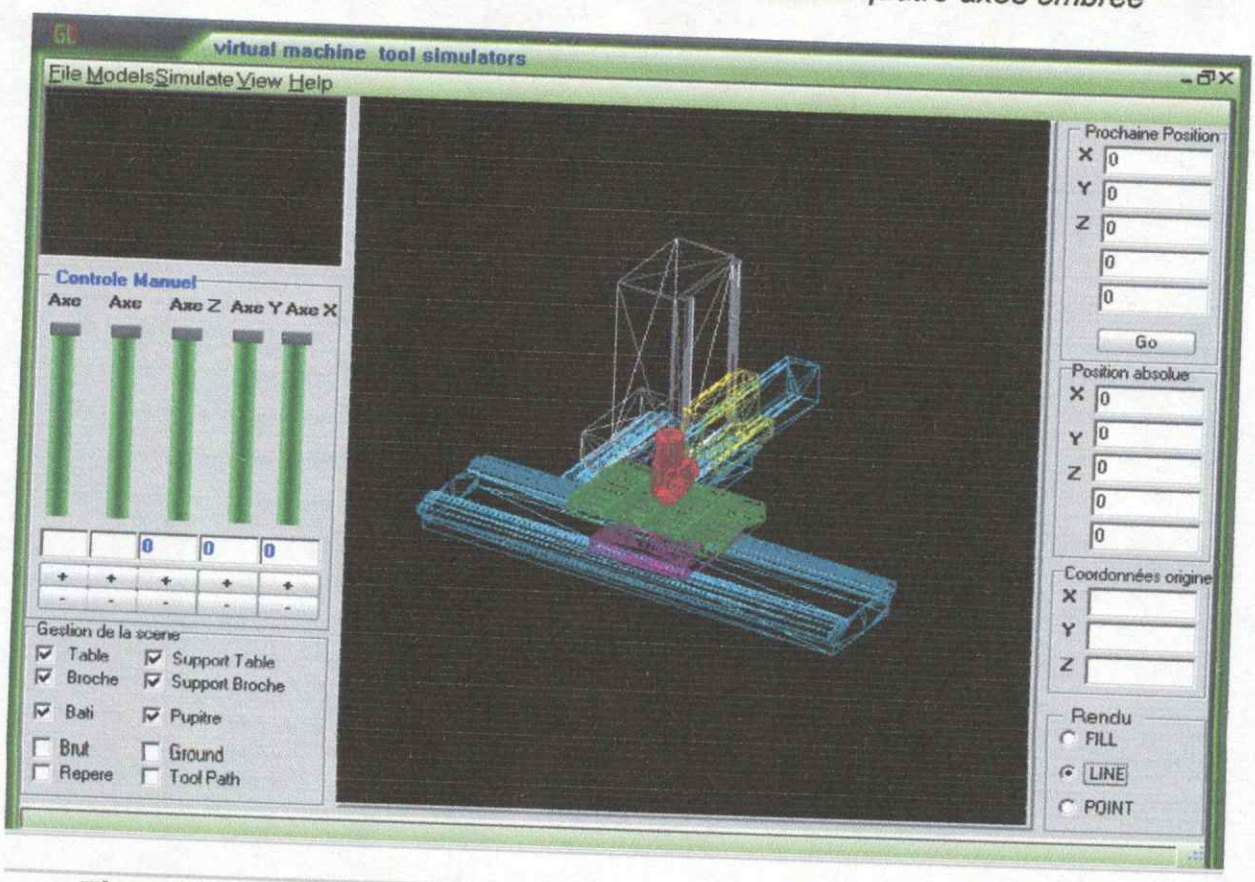


Figure V.10 :*Capture d'écran montre une fraiseuse à quatre axes en lignes*

La ***figure V.11*** est une capture d'écran, elle correspond à une version du système ou on a implémenté la cinématique directe. On peut déplacer les deux organes mobiles manuellement en glissant les trackBar associés à l'interface graphique ou en cliquant sur les boutons .

Le calcul de la cinématique directe fonctionne correctement; les interactions entre les différents organes de la machine ne présentent pas d'anomalies.

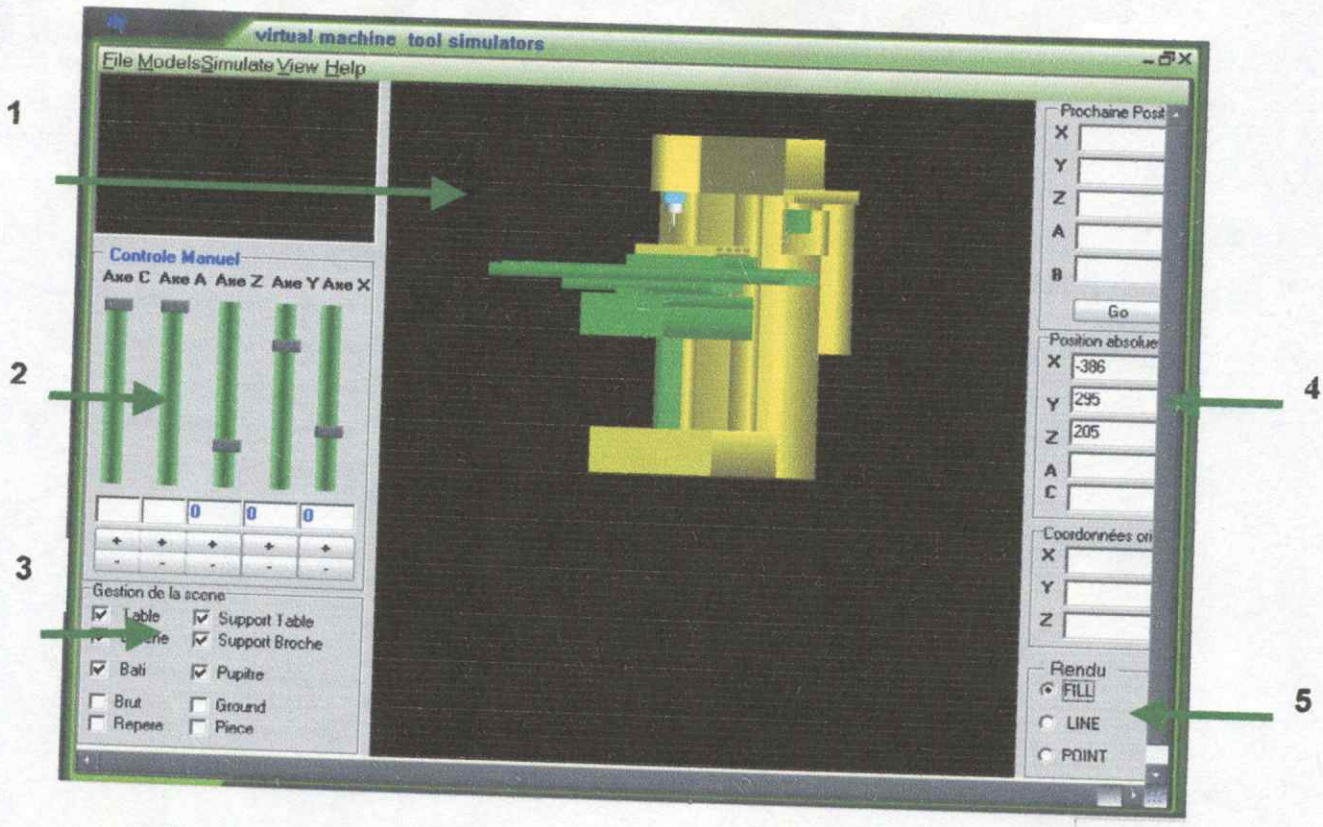


Figure V.11 :*Capture d'écran représente la cinématique directe*

1 : le panneau d'affichage de la vue de la camera
2 : le panneau du control de la machine
3 : le panneau de la gestion de la scène
4 : le panneau de l'affichage des positions
5 : le panneau de contrôle de l'apparence des objets de la scène.

II.2- La vérification de l'usinabilité :

La vérification de l'usinabilité consiste à déterminer les zones de la surface non visibles et les zones non accessibles afin de savoir si la surface est usinable ou non sur telle machine.

Pour étudier l'usinabilité nous passons par la triangulation de la surface.
La figure suivante (figure V.12) montre une surface gauche triangulée, cette surface est non usinable sans démontage sur les fraiseuses à trois axes.

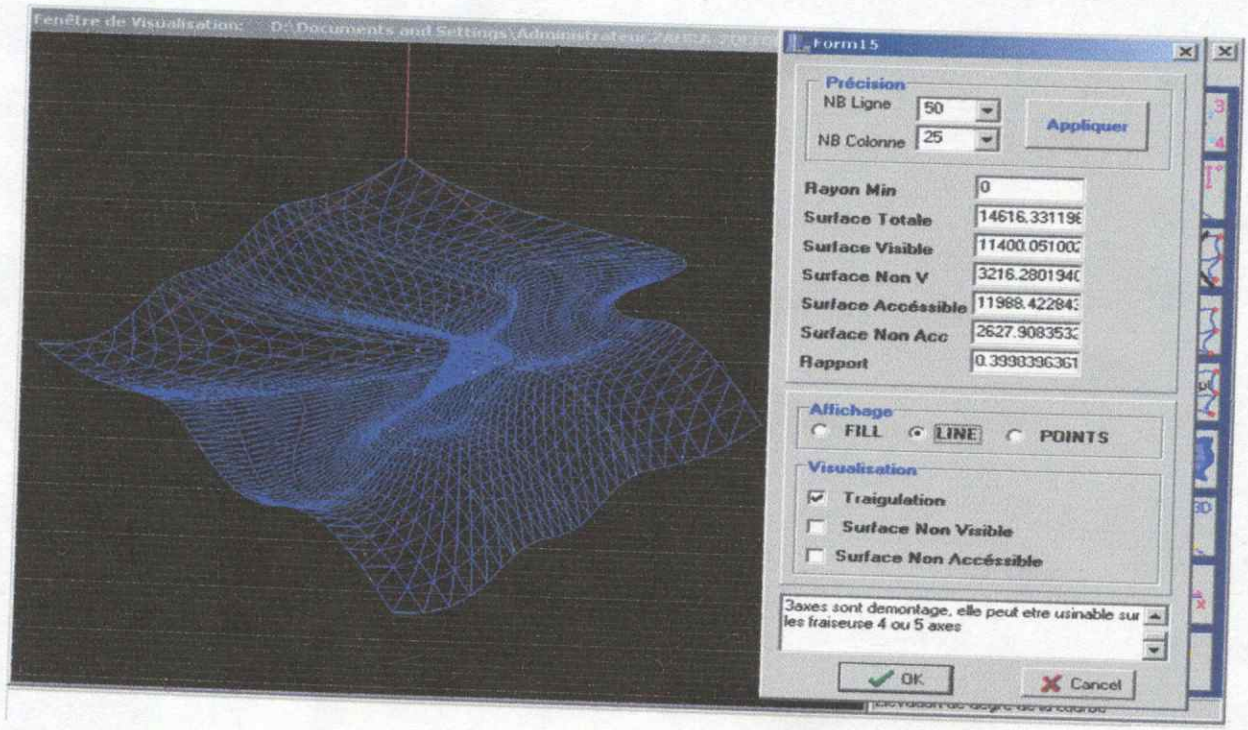


Figure V.12 : *Capture d'écran présente une surface triangulée*

La figure suivante présente une surface gauche contient des zones non visibles (jaunes) et des zones non accessibles (rouges)

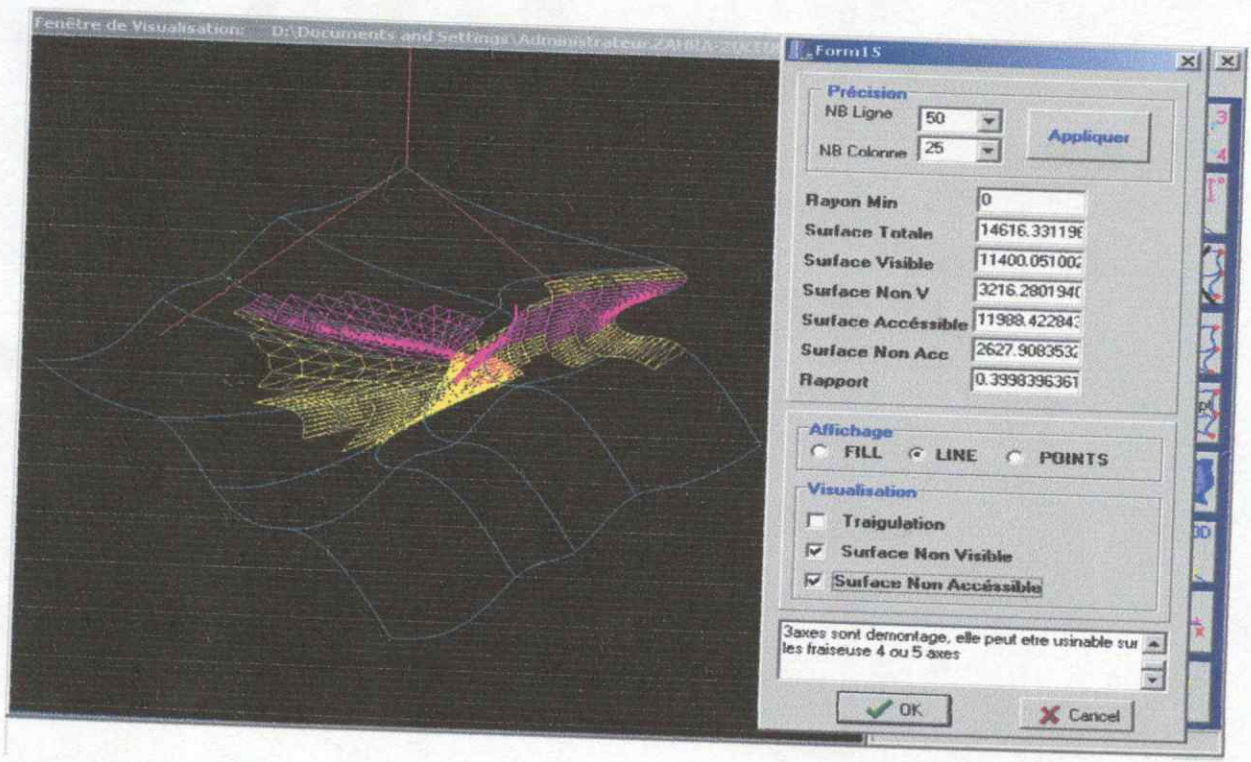


Figure V.13 : *capture d'écran présente les zones non visibles et les zones non accessibles*

II.3- La simulation de l'usinage :

la figure suivante présente la version ou on a simulé l'usinage . la simulation de l'usinage passe a les étapes suivantes :

- Faire entrer le brut à usiner (entrer les dimensions du brut ou bien choisir le brut minimal)
- Charger la zone de texte associée à la scène par l'ensemble des positions de l'outil à suivre (la trajectoire de l'outil) .
- La résolution de la cinématique inverse.

Nous sommes arrivées à résoudre la cinématique inverse pour les machine à trois axes et à cinq axes .

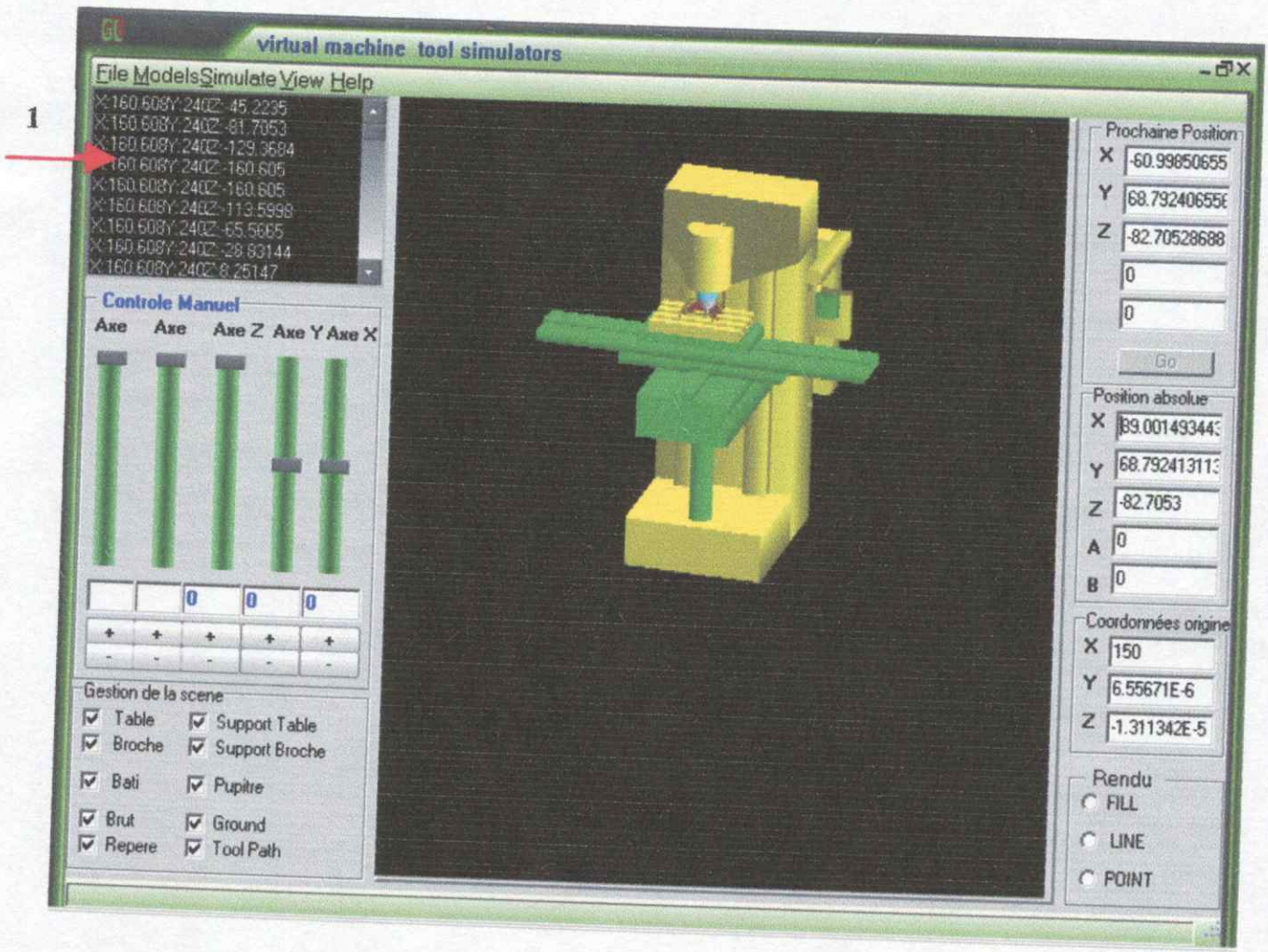


Figure V.14 : *capture d'écran présente la simulation de l'usinage*

la figure V.14 permet la visualisation de la machine , et montre la trajectoire (les positions de la trajectoire sont chargées dans la zone de texte (1)) de l'outil sur la table.

III- CONCLUSION

On a suivi une méthode itérative et incrémentale (XP), qui est basée sur les tests incrémentaux ; on peut effectuer plusieurs itérations sur le même composant logiciel, donc on peut avoir plusieurs versions jusqu'à arriver à la version finale de ce composant, pour valider une version on l'a teste ; lorsqu'on obtient des résultats satisfaisants on passe à un autre composant logiciel qui peut être réaliser après un certains nombre d'itérations et ainsi de suite jusqu'à arriver à la version complète et finale de notre système.

Cette manière de procédé nous a permis d'éviter les problèmes d'intégration de l'ensemble des composants en phase finale qui sont : le risque de la génération des erreurs importantes, des problèmes d'optimisations, le risque d'impossibilité d'assembler correctement le logiciel, des problèmes du fonctionnement de l'ensemble des composants logiciels,...etc.

CONCLUSION GENERALE

Dans cette partie :

- CONCLUSION
- LES EXTENSIONS POSSIBLES

CONCLUSION

Dans ce travail , et pour arriver à notre but qui est la virtualisation de la vérification de l'usinabilité des surfaces gauches sur des fraiseuses numériques, nous avons considéré les différentes étapes suivantes :

En premier lieu , nous avons modélisé les différentes machines du point de vue géométrique.

En deuxième lieu, nous les avons modélisé du point de vue cinématique , afin de résoudre le problème de la cinématique directe et inverse.

Vu la complexité de l'usinabilité sur des fraiseuses à cinq axes , nous n'avons pu considérer dans ce travail que l'usinabilité sur des fraiseuses à trois axes , et la simulation de l'usinage sue des fraiseuses à trois, quatre et cinq axes.

Pour arriver à virtualiser la vérification de l'usinabilité , plusieurs domaines devraient être abordés pour la maîtrise de ce sujet.

Le premier est relatif à l'étude des surfaces gauches afin d'être capable d'analyser leurs propriétés géométriques par la suite .

Comme ses surfaces ne sont usinables que sur des machines outils à commande numérique , l'étude des machines outils à commande numérique en général et les fraiseuses numériques est également imposé.
Nous avons su à travers notre étude qu'il y a trois classes des fraiseuses : celles à trois axes et à quatre axes et les fraiseuses à cinq axes , l'usinage sur ces machines est un aspect également qu'il fallait maîtrisé.

Ces études nous ont permis d'aboutir à la partie conception dont le but est la vérification de l'usinabilité des surfaces gauches sur les fraiseuses numériques.

Nous avons utilisé l'eXtreme Programming (XP) qui est une méthode agile de gestion de projets informatiques , elle permet un développement rapide de notre système.

Et pour l'implémentation de notre logiciel , nous avons utilisé les concepts de l'orienté objet , qui ont l'avantage de simplifier l'apparence du programme et faciliter ses tests , sa mise au point et sa maintenance et réduisent les risques.

Notre simulateur est programmé en BUILDER C++6 sous environnement Windows et pour une meilleure visualisation et animation de la scène à simuler , nous avons eu recours à la bibliothèque graphique OPEN GL.

On peut dire que le but de notre projet est atteint , notre logiciel peut vérifier l'usinabilité d'une certaine surface gauche sur une certaine configuration de machine , et meme simuler l'usinage de cette surface sur cette configuration de machine avant de passer à l'usinage de cette surface sur des machines réelles , par conséquent les risques de collisions et d'avoir des pièces mal usinées est minimisé.

Les extensions possibles :

- Implémenter un interpréteur capable d'interpréter des fichier G_code ISO standard , et l'améliorer par la suite en ajoutant des commandes spécifiques de certaines machines.

- Faire implémenter des algorithmes de l'usinabilité sur des fraiseuses à cinq axes.

- Détecter et éviter les collisions entre l'outil et la pièce .

- Détecter et éviter les collisions entre l'outil et la machine.

- Intégrer l'aspect dynamique en faisant une modélisation des machines plus détaillée.
- ....

BIBLIOGRAPHIE

[1] Jean Danseau avec la collaboration de Clement Fortin & Alain Boryer Christian Bellefleur ,« ***Elements de CAO*** », Département de génie mécanique, Ecole polytechnique de Montereal.

[2] Gerald Farin, « ***Courbes et surfaces pour la CGAO*** », Edition Masson, France 1992.

[3] C.K Shene , « ***Introduction to computing with geometry note*** », Departement of computer science , Michigan technological , university 1997.

[4] Philippe Lavoir , « ***An introduction to NURBS*** » , Janvier 1997.

[5] Jean-Claude Léon. « ***Modélisation et construction des surfaces pour la CFAO*** ». Edition Hermès, France 1991.

[6] Emmanuel Duc, Emmanuel Lefur. " ***Machines-outils à commande numérique, structure, modélisation et réglage*** ". Mémoire de DEA à l'école normale supérieur de Cachan, France, septembre 1997.

[7] Philipe Marin. " ***Machine à commande numérique*** ". Edition Hermès, France 1996.

[8] Claude Marty, Claude Cassagnes, Philipe Marin. " ***La pratique des machine à commande numérique*** ". Edition Hermès, France 1993.

[9] ***"Techniques de l'ingénieur"*** , Traité de mécanique et de chaleur, BT2.

[10] « ***Encyclopédie des sciences industrielles QUILLET*** », librairie Aristide QUILLET.

[11] Emmanuel DUC. " ***Usinage de formes gauches; contribution à l'amélioration de la qualité des trajectoires d'usinage*** ". Thèse de doctorat à l'école normale supérieur de Cachan, France 1998.

[12] Alain Bernard , Lavoisier « ***Fabrication assistée par ordinateur*** »,2003

[13] D.C.H. Yang*, Z.Han, « **Interference detection and optimal tool selection in 3-axis NC machining of free –form surfaces**", CAD/CAM laboratory , department of Mechanical Aerospace Engeneering, University of California, Los Angelos,1999.

[14] Grady Booch, James Rumbaugh, Ivar Jacobson, ***« Le guide de l'utilisateur UML»***, EYROLLES 2003

[15] J. –P Gourret , « **Modélisation d'images fixes et animées** », Masson 1994

[16] Michael Dixon, Melissa Lima. "***OpenGL programming guide***". Addisson-Wesley Publishing Company, 1997.

[17] Benoît Piranda, "***Initiation à OpenGL / GLUT***". Université de Marne, La.Vallée, équipe de Synthèse d'images, France, Décembre 2001.

Annexe A :

Le langage UML

(Unified Modeling Language)

A

I- INTRODUCTION

UML (Unified Modeling Language) est un langage graphique conçu pour représenter, spécifier , construire et documenter les artefacts d'un système à dominante logicielle Il permet d'écrire avec un langage standardisé les plans d'élaboration et de construction de logiciels . Il prend en compte aussi bien des éléments conceptuels tels que les processus d'entreprises et de fonctions du système , que des éléments concrets tels que les classes écrites dans un langage de programmation , les schémas de bases de données et les composants logiciels réutilisables.

La notation UML provient de la fusion des notations de **BOOCH**, **OMT**, **OOSE** et d'autres notations. Les grands traits des objets s'articulent autour :

- des notions de classe et d'association décrites par JAMES RUMBAUGH dans la méthode OMT (*Object Modeling Technique*),
- de partition en sous-systèmes avec la méthode mise au point par GRADY BOOCH (méthode BOOCH),
- et autour de l'expression des besoins à partir de l'étude de l'interaction entre l'utilisateur et le système : cas d'utilisation (*uses cases*) d'IVAR JACOBSON (méthode OOSE).

1. HISTORIQUE D'UML

1994 : Rumbaugh (OMT) rejoint Booch (OOD) chez Rational Software, but : créer une *méthode* en commun.

1995 : présentation v0.8.

1995 : Arrivée de Jacobson (OOSE).

1996 : *Langage* unifié UML 0.9 (Unified Modeling Language).

1997 : présentation de UML à l'OMG (Object Management Group) UML 1.1 adopté par la plupart des compagnies.

1998 : UML 1.2.

1999 : UML 1.3.

2003 : standardisation par OMG de UML 2.0.

Le diagramme suivant **(Figure B.1)** montre les différentes étapes de l'évolution du langage.

II- OBJECTIFS D'UML

- Montrer les limites d'un système et ses fonctions principales à l'aide des cas d'utilisation et des acteurs.
- Illustrer les réalisations de Cas d'Utilisations à l'aide de diagrammes d'interaction (de séquences).
- Représenter la structure statique d'un système à l'aide de diagrammes de classes, association, contraintes.
- Modéliser la dynamique, le comportement des objets à l'aide de diagrammes états/transitions.
- Révéler l'implantation physique de l'architecture avec des diagrammes de composants et de déploiement.
- Un langage utilisable par l'homme et la machine : permettre la génération automatique de code.

III- LE MODELE CONCEPTUEL D'UML

Le modèle conceptuel d'UML comprend les notions de base génériques du langage. Il définit trois concepts de base:

- Les éléments, qui sont les abstractions essentielles à un modèle ;
- Les relations, qui constituent des liens entre ces éléments ;
- Les diagrammes, qui regroupent des éléments et des liens au sein de divers ensembles.

III.1- LES ELEMENTS :

Il existe quatre types d'éléments dans UML :

- Les éléments structurels (classe, interface, collaboration,...) ;
- Les éléments comportementaux (interaction, automate à états finis) ;

- Les éléments de regroupement (package) ;
- Les éléments d'annotation (note).

III.2- LES PACKAGES :

Les packages sont des mécanismes d'ordre général qui permet d'organiser les éléments en groupes. Ils permettent de définir des sous-systèmes formés d'éléments ayant entre eux une certaine logique. Leurs caractéristiques sont les suivantes :

- Ils regroupent des éléments de modélisation selon des critères purement logiques ;
- Ils permettent d'encapsuler des éléments de modélisation par l'intermédiaire d'interfaces ;
- Ils permettent de structurer un système en catégories ou sous-systèmes ;
- Ils servent de composants réutilisables dans la conception d'un logiciel.

L'importation permet aux éléments d'un package d'accéder aux éléments d'un autre package. Cette relation est à sens unique et est représentée par une relation de dépendance « import ».

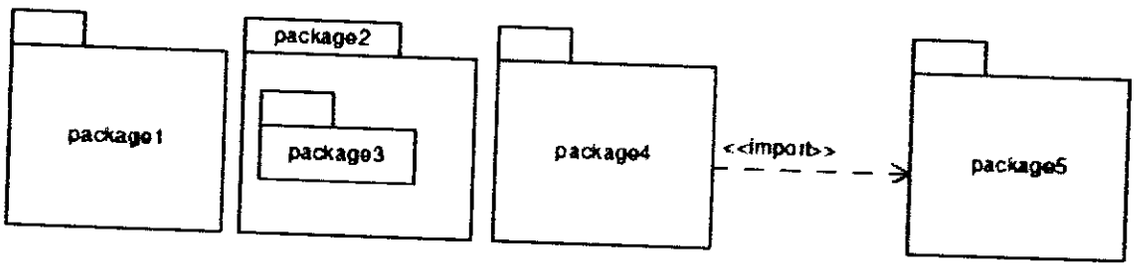


Figure B.2

III.3- LES CLASSES :

Une Classe est une représentation d'un ensemble d'éléments partageant les mêmes attributs, les mêmes opérations, les mêmes relations et les mêmes sémantiques. Graphiquement, une classe décrite en UML peut être plus ou moins précise **(Figure B.3)**.

La visibilité d'une caractéristique détermine si d'autres éléments peuvent l'utiliser. Trois niveaux de visibilité sont possibles pour les attributs et les opérations :

- ' + ' : ***visibilité public***. La caractéristique peut être utilisée par n'importe quelle instance ayant une visibilité sur les instances de la classe complétée.
- ' - ' : ***visibilité private***. La caractéristique ne peut être utilisée que par des instances de la classe elle même
- ' # ' : ***visibilité protected***. La caractéristique ne peut être utilisée que par des instances de la classe elle même ou bien par les descendants directs de cette classe.

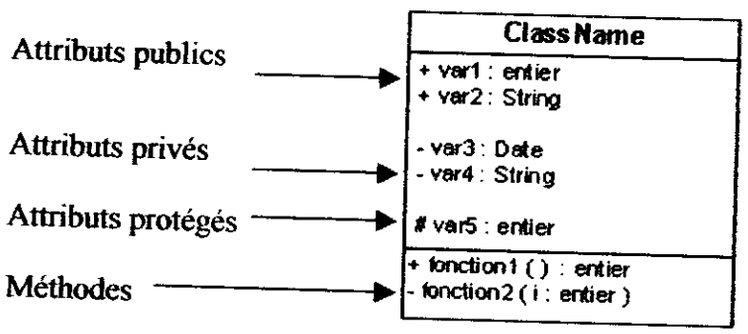


figure B.3

III.3.1- LES CLASSES ABSTRAITES :

Une Classe abstraite est une classe ne pouvant pas être instancier directement. Une telle classe sert de spécification pour des objets instances de ses sous-classes. **(Figure B.4)**

III.3.2- LES INTERFACES :

Une Interface: décrit un contrat d'une classe ou d'un composant sans en imposer l'implémentation. Une interface ne décrit aucune structure ni aucune implémentation. Elle ne peut donc pas contenir d'attributs, ni de méthodes fournies sous la forme d'une implémentation. **(Figure B.5)**

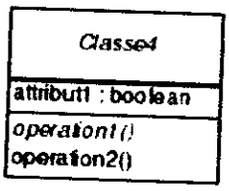


Figure B.4

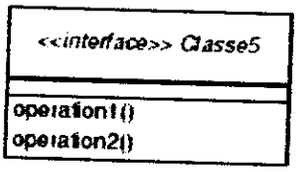


Figure B.5

III.3.3- LES RELATIONS :

Il existe quatre types de relations dans UML :

- L'association;
- La dépendance;
- La généralisation ;
- L'implémentation ;
- L'agrégation ;
- La composition.

III.3.4- L'ASSOCIATION :

La relation d'association est une relation structurelle précisant que les objets d'un élément sont reliés aux objets d'un autre élément.

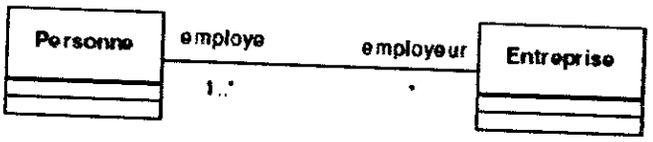


Figure B.6

III.3.5- LA DEPENDANCE :

La relation de dépendance est une relation sémantique entre deux éléments selon laquelle un changement apporté à l'un peut affecter la sémantique de l'autre. **(Figure B.7).**

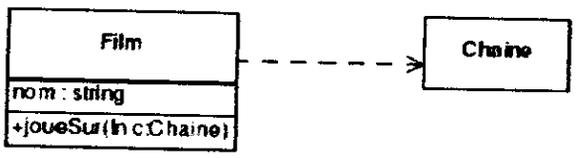


Figure B.7

III.3.5- LA GENERALISATION :

La relation de généralisation est une relation entre un élément général et un élément dérivé de celui-ci, mais plus spécifique (désigné par sous élément ou élément fils).Le plus souvent, la relation de généralisation est utilisée pour représenter une relation d'héritage. **(Figure B.8).**

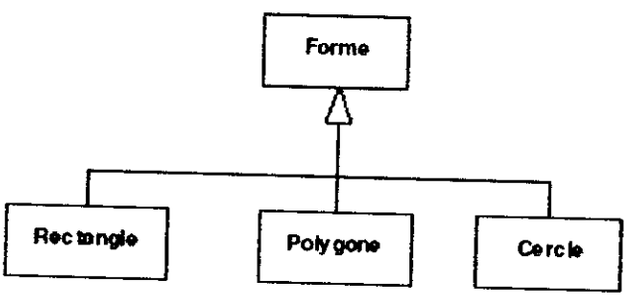


Figure B.8

III.3.6- L'IMPLEMENTATION :

La relation d'implémentation: relation entre une classe et une interface spécifiant que la classe implémente les opérations définies par l'interface. **(Figure B.9).**

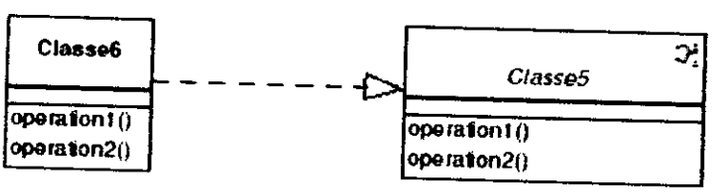


Figure B.9

III.3.7- L'AGREGATION :

La relation d'agrégation représente une relation de type "ensemble / élément"**(Figure B.10).** Une agrégation peut notamment (mais pas nécessairement) exprimer :

- qu'une classe (un "élément") fait partie d'une autre ("l'agrégat"),
- qu'un changement d'état d'une classe, entraîne un changement d'état d'une autre,
- qu'une action sur une classe, entraîne une action sur une autre.

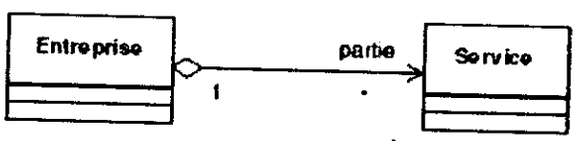


Figure B.10

III.3.7- LA COMPOSITION :

La relation de composition: relation d'agrégation forte. Les ´parties´ sont crées et détruites en même temps que le ´tout´. **(Figure B.11).**

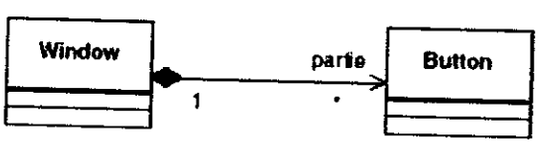


Figure B.11

IV- DIFFERENTS TYPES DE DIAGRAMMES UML

Un diagramme est une représentation graphique d'un ensemble d'éléments et de relations qui constituent un système. UML définit neuf types de diagrammes divisés en deux catégories :

- Vues statiques :

- **Diagrammes de classes :** représentent la structure statique en termes de classes et de relations.
- **Diagrammes d'objets :** représentent les objets et leurs liens, instances des classes et de leurs relations ; lorsque les interactions entre objets sont précisées sur les liens, ces diagrammes correspondent à une forme de diagramme de collaboration.
- **Diagrammes de cas d'utilisation :** représentent les fonctions du système du point de vue des utilisateurs.
- **Diagrammes de composants :** représentent les composants physiques d'une application.
- **Diagrammes de déploiement :** représentent le déploiement des composants sur les dispositifs matériels.

- Vues dynamiques :
  - **Diagrammes de séquence :** offrent une représentation temporelle des objets et de leurs interactions.
  - **Diagrammes de collaboration :** offrent une représentation spatiale des objets, de leurs liens et interactions.
  - **Diagrammes d'états-transitions :** représentent les changements d'états, et les comportements associés, pour un classificateur ou une méthode.
  - **Diagrammes d'activités :** représentent le comportement d'une méthode, d'un cas d'utilisation ou d'un processus métier.

V- NOTATIONS UML

V.1 LES DIAGRAMMES STATIQUES :

V.1.1- LES DIAGRAMME DES CAS D'UTILISATION (USE CASE DIAGRAM) :

Ils sont constitués d'acteurs et de cas d'utilisation. **(Figure B.12).**

Un acteur Un acteur est une entité externe qui interagit avec notre système (une personne ou un système *extérieur*).

Les cas d'utilisations décrivent les fonctionnalités fournies par le système à un acteur. Ils décrivent une famille de scénarios incluant les cas d'erreur. Un scénario décrit une séquence particulière de messages dans le cas d'utilisation.

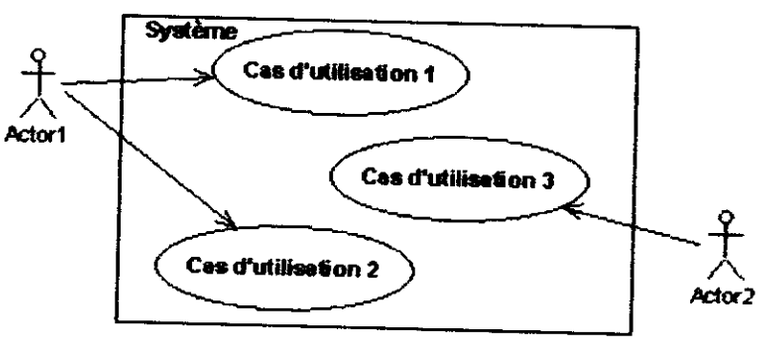


Figure B.12

Il existe trois types de relations entre cas d'utilisation :

- La relation de généralisation : le cas d'utilisation enfant est une spécialisation du cas d'utilisation parent. **(Figure 8.13)**

- La relation d'inclusion (*include*) : le cas d'utilisation source comprend également le comportement de son cas d'utilisation destination. Cette relation a un caractère obligatoire telle que la réalisation de l'un nécessite la réalisation de l'autre (à la différence de la généralisation) et permet ainsi de décomposer des comportements partageables entre plusieurs cas d'utilisation différents. **(Figure 8.13)**

- Le relation d'extension (*extends*) : le cas d'utilisation source ajoute son comportement au cas d'utilisation destination. L'extension peut être soumise à condition. Cette relation permet de modéliser des variantes de comportement d'un cas d'utilisation. **(Figure 8.13)**

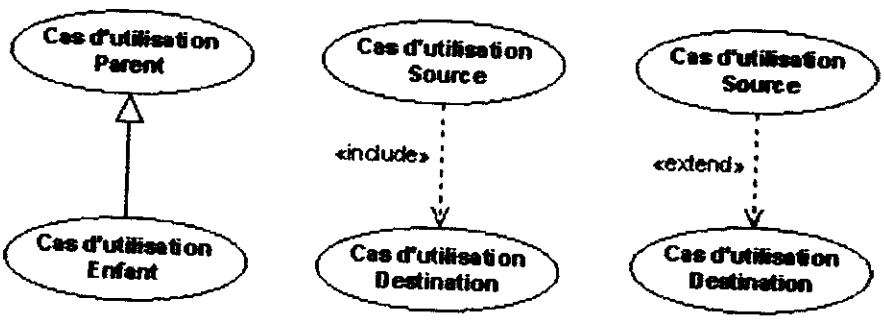


Figure 8.13

V.1.2- LES DIAGRAMMES DE CLASS (CLASS DIAGRAM):

Ils sont principalement constitués d'éléments structuraux tels que les class, les interfaces et les modules (*package*). **(Figure B.14)**

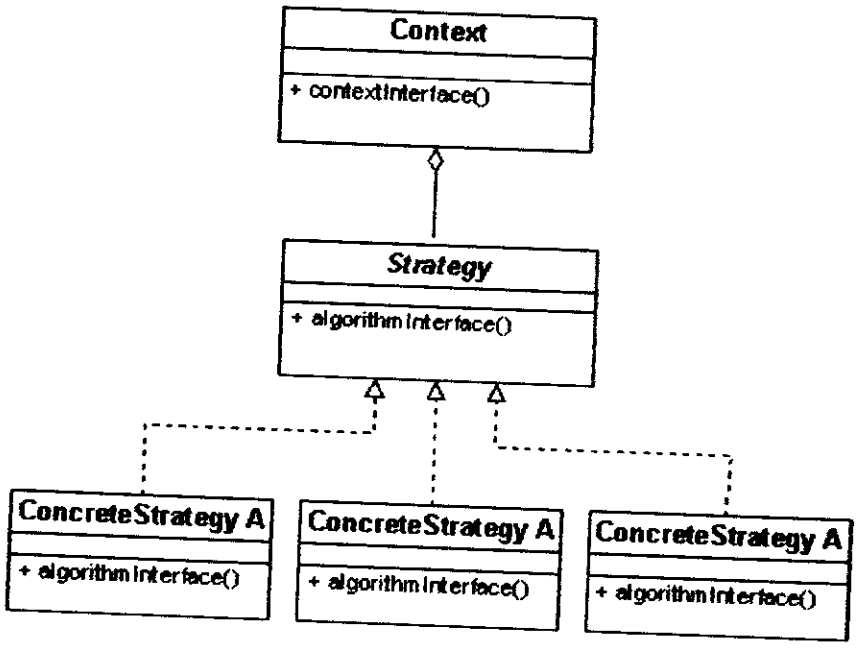


Fig. 14

V.1.3- LES DIAGRAMMES D'OBJETS (OBJECT DIAGRAM):

Ils représentent les objets et leurs relations et correspondent à des diagrammes de collaboration simplifiés, sans représentation des envois de messages. **(Figure B.15)**

Figure 8.15

V.1.4- LES DIAGRAMMES DE COMPOSANTS (COMPONENT DIAGRAM):

Ils permettent de décrire les composants et leurs dépendances dans leur environnement d'implémentation. **(Figure B.16).**

Un Composant: élément physique qui représente une partie implémentée d'un système. Il peut être du code, un script, un fichier de commandes, etc. Ils présentent un ensemble d'interfaces. Les composants peuvent être organisés en sous-sytèmes de packages permettant de masquer la complexité, par l'encapsulation des détails d'implémentation. **(Figure 8.16).**

Interfaces d'un composant: élément définissant le comportement offert à d'autres composants

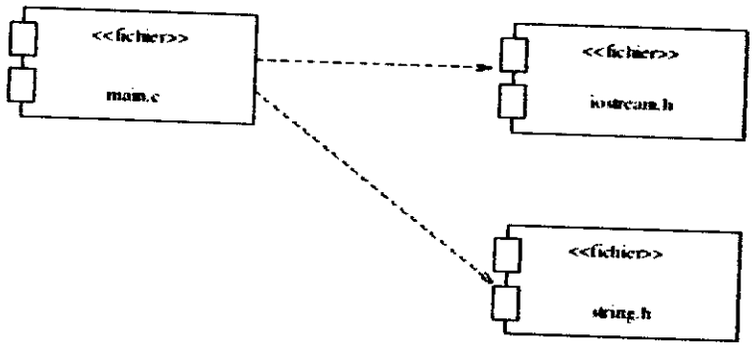


Figure B.16

V.1.5- LES DIAGRAMMES DE DEPLOIEMENT (DEPLOYMENT DIAGRAM):

Ils représentent le déploiement des composants sur les dispositifs matériels représentés par des noeuds. **(Figure B.17).**

Ce type de diagramme est utilisé principalement pour la modélisation de trois types de systèmes : les systèmes embarqués, les systèmes client/serveur et les systèmes totalement répartis. Deux notations sont possibles pour montrer qu'un composant réside sur un noeud :

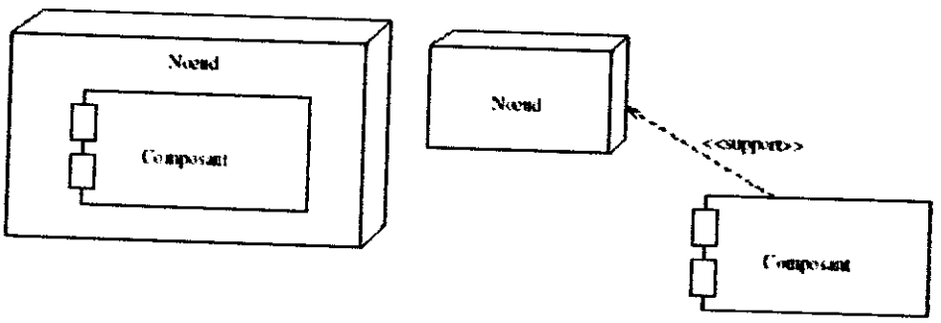


Figure 8.17

V.2- LES DIAGRAMMES DYNAMIQUES :

V.2.1- LES DIAGRAMMES D'ACTIVITES (ACTIVITY DIAGRAM) :

Les diagrammes d'activités représentent le comportement d'une opération comme : une opération, un cas d'utilisation, une classe ou une méthode...etc. **(Figure B.18).**

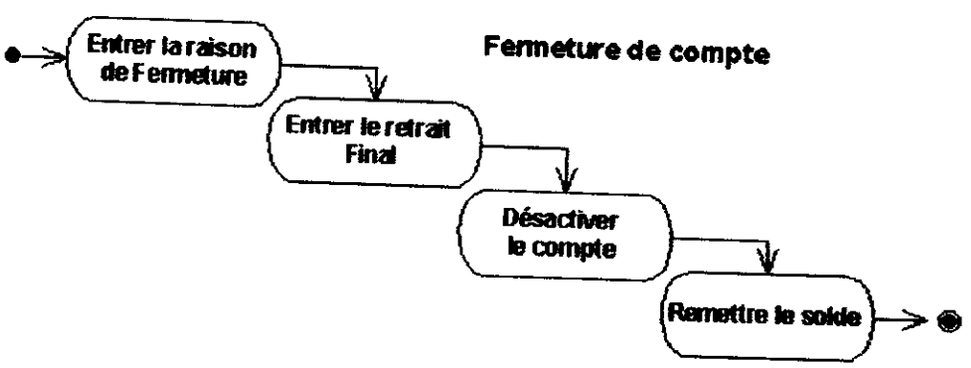


Figure B.18

V.2.2- LES DIAGRAMMES DE SEQUENCES (SEQUENCE DIAGRAM) :

Ils donnent la représentation temporelle des objets et de leurs interactions. La représentation se concentre sur l'expression des interactions et non pas sur l'état ou le contexte des objets. Ce type de diagramme est usuellement utilisé pour illustrer les diagrammes de cas d'utilisation.

Les Interactions modélisent un comportement dynamique entre objets. Elles se traduisent par l'envoi de messages entre objets. Un diagramme de séquence représente une interaction entre objets, en insistant sur la chronologie des envois de messages.

Un Message est une représentation d'une communication au cours de laquelle des informations sont échangées : appels de procédures, signaux,...etc. Ils sont représentés par des flèches et leur ordre est donné par leurs positions sur la ligne de vie. **(Figure B.19).**

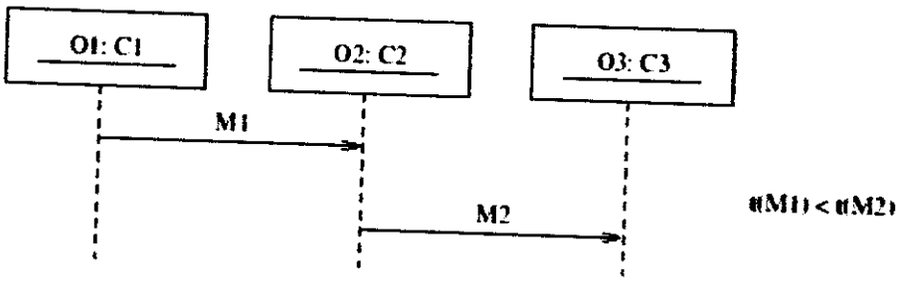


Figure B.19

Dans un diagramme de séquence, les objets sont associés à une ligne de vie. La dimension verticale de celle-ci représente l'écoulement du temps (du haut vers le bas). Notons que la disposition des objets sur l'axe horizontal n'est pas importante dans ce type de diagrammes. **(Figure B.20).**

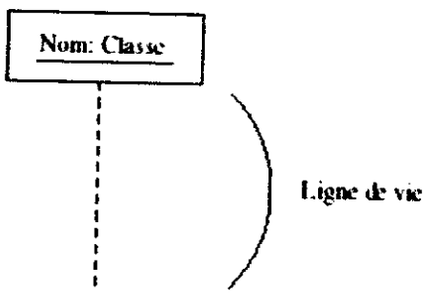


Figure B.20

Une Période d'activation correspond au temps pendant lequel un objet effectue une action, soit directement, soit par l'intermédiaire d'un autre objet. **(Figure 8.21).** Dans la Figure **(Figure 8.22)** O1 active O2. La période d'activation de O1 recouvre celle de O2. Dans le cas d'un appel de procédure, O1 est bloqué jusqu'à ce que O2 lui redonne la main.

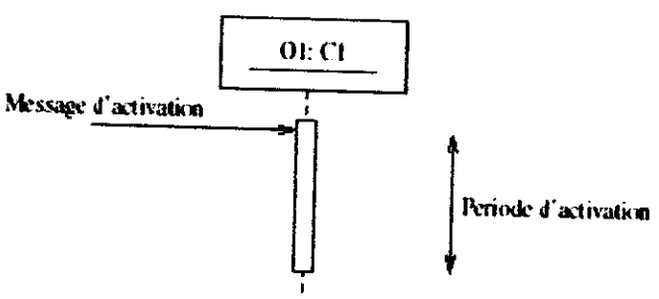


Figure B.21

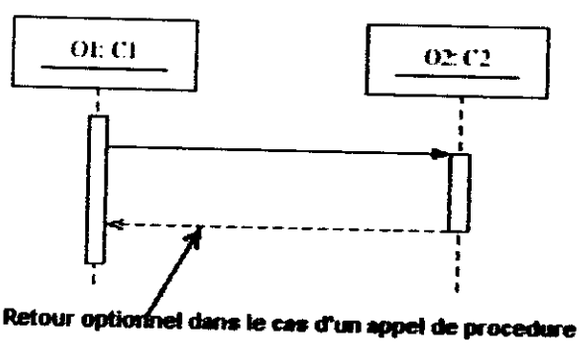


Figure B.22

Il existe deux catégories d'envoi de messages : *Les flots de contrôle à plat* (simple progression vers la prochaine étape d'une séquence) et *les flot de contrôle emboîtés* (la séquence emboîtée doit se terminer pour que la séquence englobante reprenne le contrôle).

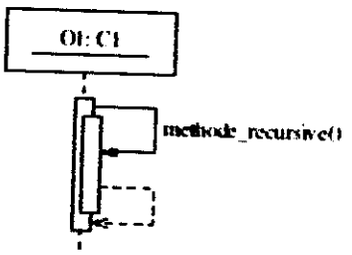


Figure B.23

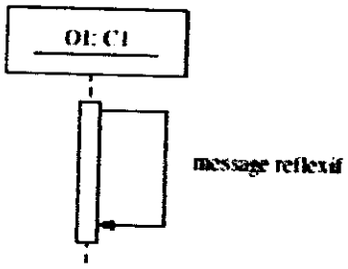


Figure B.24

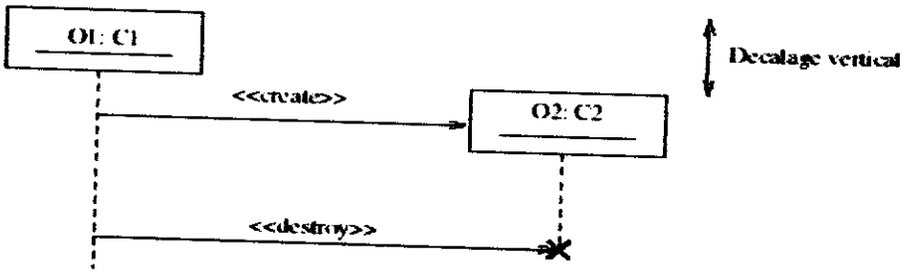


Figure B.25

V.2.3- LES DIAGRAMMES D'ETATS-TRANSITIONS (STATECHART DIAGRAM) :

Ils représentent le comportement d'un classificateur ou d'une opération en terme d'états.(**Figure B.26)**

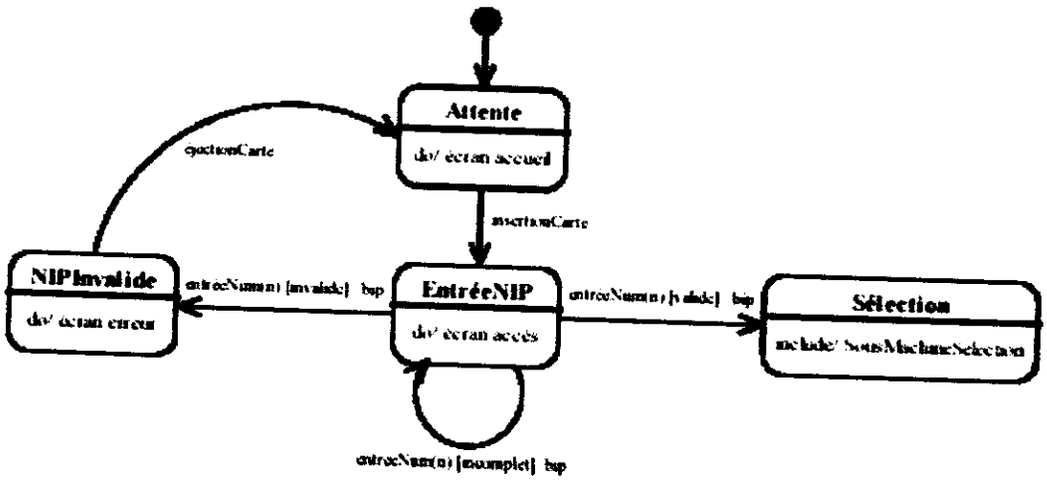


Figure B.26

V.2.4- LES DIAGRAMMES DE COLLABORATION (COLLABORATION DIAGRAM) :

C'est une représentation spatiale des objets, des liens et des interactions. **(Figure B.27)**

Figure B.27

La syntaxe d'un message est la suivante :

[pré "/"] [["["cond"]"] [séq] ["*"["||"]["["iter"]"]]] ":"] [r ":="] msg"("[par]")"

pré : prédécesseurs (liste de numéros de séquence de messages séparés par une virgule ; voir aussi "**séq**").Indique que le message courant ne sera envoyé que lorsque tous ses prédécesseurs le seront aussi (permet de synchroniser l'envoi de messages).

cond : garde, expression booléenne permet de conditionner l'envoi du message, à l'aide d'une clause exprimée en langage naturel.

séq : numéro de séquence du message.
Indique le rang du message, c'est-à-dire son numéro d'ordre par rapport aux autres messages. Exemple : l'envoi du message 1.3.5 suit immédiatement celui du message 1.3.4 et ces deux messages font partie du flot (de la famille de messages) 1.3.

iter : récurrence du message.Permet de spécifier en langage naturel l'envoi séquentiel (ou en parallèle, avec "||") de messages. Notez qu'il est aussi possible de spécifier qu'un message est récurrent en omettant la clause d'itération (en n'utilisant que "*" ou "*||").

r : valeur de retour du message.
Permet d'affecter la valeur de retour d'un message, pour par exemple la

retransmettre dans un autre message, en tant que paramètre.

msg : nom du message.

par : paramètres (optionnels) du message.

VI- LIMITATIONS D'UML

VI.1- Une notation et pas une méthode

A l'opposé de Merise, UML n'est basé sur aucune base mathématique et logique55. Il n'existe donc aucun contrôle contre des modélisations absurdes ou 'contres natures'.

Le diagramme, ci-dessous, pourtant simpliste au possible, est un exemple flagrant d'un schéma UML absurde **(Figure B.28)** :

Ici 'Fils' hérite de 'Père' (flèche au niveau de 'Père') mais 'Père' hérite aussi de 'Fils' ce qui est une négation totale du principe de programmation Orienté Objet. En effet, en POO, il ne peut exister d'héritage 'à double sens'.

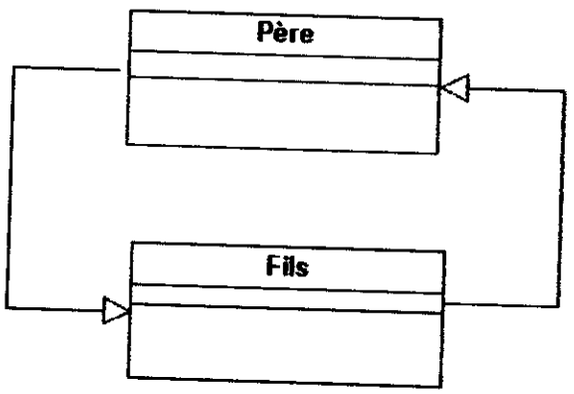


Figure B.28

En ce sens, UML est 'dangereux' car il ne pose aucun contrôle. Il laisse au développeur la capacité de produire des conceptions totalement pauvres de sens ou fausses.

VII- CONCLUSION

UML fournit un moyen standard de visualiser, spécifier, construire, documenter et communiquer les composants d'un système logiciel. C'est une notation et non une méthode, un moyen d'expression qui peut être utilisé dans le cadre d'un processus logiciel (méthode) comme le RUP (*Rational Unified Process*).

Ainsi, nous pouvons exprimer les différentes classes et leur type de relations, les rôles
qu'ils occupent, les transitions possibles entre leurs divers états, l'ordre relatif des événements possibles, la dépendance entre les composantes du logiciel ainsi que la distribution des instances au niveau physique.

L'objectif est d'obtenir des applications logicielles robustes, résistantes et évolutives. UML est suffisamment général pour être employé dans tous les domaines informatiques.

B

Annexe B :

La méthode de conception XP

(eXtreme Programming)

EXTREME PROGRAMMING

L'Extreme Programming (XP) est une méthode agile de gestion de projets informatiques , dont l'objectif est de permettre de gérer des projets de manière simple et efficace. Cette méthode a été inventée par KENT BECK et WARD CUNNINGHAM. L'Extreme Programming est née officiellement en octobre 2000 avec le livre « Extreme Programming Explained » de KENT BECK.

L'Extreme Programming (XP) est un processus de développement logiciel, c'est-à-dire un ensemble de pratiques destinées à organiser le travail d'une équipe de développement. Ces pratiques se focalisent sur la construction proprement dite du logiciel, en aval des phases préparatoires d'études d'opportunité ou de faisabilité.

Comme toutes les méthodes de développement, l'Extreme Programming propose un cadre pour l'ensemble des aspects du projet logiciel, depuis l'analyse des besoins jusqu'aux tests, en passant par la conception. Mais à la différence des processus prédictifs, recourant généralement à UML (même si XP utilise aussi parfois UML comme support de communication), XP ne se fonde pas sur la définition exhaustive et précoce des besoins ; elle parie plutôt, à partir d'un ensemble de règles strictes, sur la souplesse et la mise en valeur du « capital humain ».

XP est l'un des principaux représentants d'une famille émergente de processus : les processus dits "agiles", qui se démarquent des démarches traditionnelles en mettant l'accent sur le travail d'équipe et la réactivité. L'heure est à l'économie et à l'efficacité:

« Quelles activités pouvons-nous abandonner tout en produisant des logiciels de qualité ? »

Ou encore :

« Comment mieux travailler avec le client pour nous focaliser sur ses besoins les plus prioritaires et être aussi réactifs que possible ? »

XP propose une réponse originale à ces questions, avec un ensemble de pratiques organisées autour des principes suivants :

- **Le client (maîtrise d'ouvrage) pilote lui-même le projet**, et ce de très près grâce à des cycles itératifs extrêmement courts (1 ou 2 semaines).
- **L'équipe livre très tôt dans le projet** une première version du logiciel, et les livraisons de nouvelles versions s'enchaînent ensuite à un rythme soutenu pour obtenir un feedback maximal sur l'avancement des développements.
- **L'équipe s'organise elle-même pour atteindre ses objectifs**, en favorisant une collaboration maximale entre ses membres.
- **L'équipe met en place des tests automatiques** pour toutes les fonctionnalités qu'elle développe, ce qui garantit au produit un niveau de robustesse très élevé.
- **Les développeurs améliorent sans cesse la structure interne du logiciel** pour que les évolutions y restent faciles et rapides.

LE CYCLE DE L'EXTREME PROGRAMMING

Le cycle de développement XP

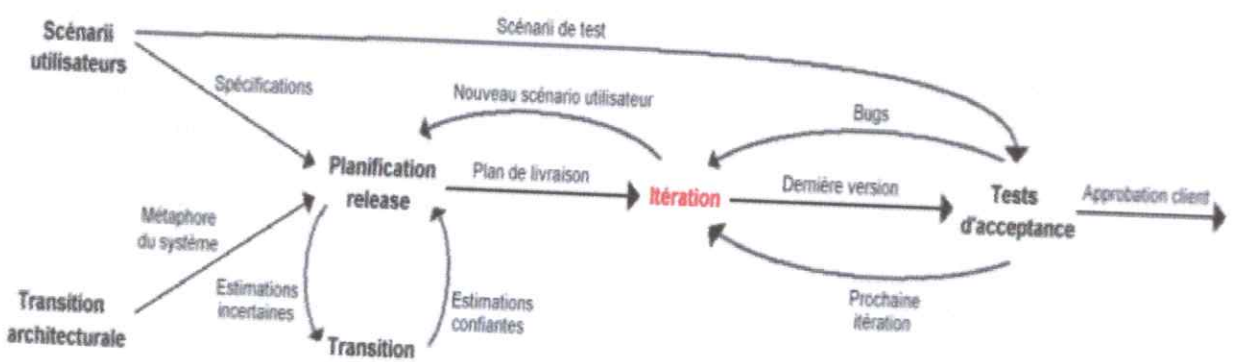


figure A.1

LES PRATIQUES XP

Les pratiques XP sont pour la plupart des pratiques de bon sens utilisées par des développeurs et des chefs de projets expérimentés. On retrouve par exemple les tests unitaires automatisés, les livraisons fréquentes ou encore les relectures de code.

La première nouveauté d'XP consiste à pousser ces pratiques à l'extrême (d'où le nom de la méthode), ou comme le disent ses auteurs "tourner tous les boutons jusqu'à 10 !" L'équipe utilise des cycles de développement d'une ou deux semaines, les développeurs écrivent des tests unitaires pour chaque classe, ils se livrent à une relecture de code permanente via le travail en binômes, etc.

La seconde nouveauté d'XP consiste à organiser ces pratiques en un tout cohérent, de sorte que chaque pratique renforce les autres. Il en résulte une méthode complète, qui couvre tous les aspects du développement - de la relation avec le client jusqu'à l'écriture du code, en passant par les plannings et l'organisation de l'équipe.
Voici les principaux éléments du fonctionnement d'XP :

- **PILOTAGE DU PROJET**

➢ ***Le rôle de « client XP »***

Le pilotage du projet est assuré par un membre spécifique de l'équipe projet : le « client ». Le client détermine les fonctionnalités du logiciel, gère les priorités, définit les spécifications précises du produit – en somme, ce rôle correspond à ce que nous nommons en France la maîtrise d'ouvrage du projet. Dans un projet XP, ce représentant de la maîtrise d'ouvrage rejoint le projet à plein temps.

Le choix de cette personne dépend très fortement du type de projet. Dans les contextes les plus simples, il pourra s'agir d'une personne qui est à la fois donneur d'ordre et utilisateur final. Dans d'autres contextes, le client pourra être le représentant d'un groupe plus large réunissant un comité de direction, des architectes, etc.

➢ ***La phase initiale d'exploration***

Le projet démarre par une phase d'exploration volontairement très courte (typiquement un mois maximum), qui a pour triple objectif de définir le contenu fonctionnel de l'application, établir un premier plan de développement pour le projet, et produire la toute première version du logiciel.

Au cours de cette phase, le client explore et définit le contenu fonctionnel qu'il souhaite voir implémenté dans le produit. Il établit ainsi une liste de fonctionnalités, qui prennent en XP la forme de « scénarios client ».
Un scénario client décrit en quelques mots un besoin particulier du client. Par exemple, pour un logiciel de gestion de carnet d'adresses le client pourrait écrire les scénarios suivants :

- "Je rentre un nom ou prénom, et le logiciel affiche la liste de toutes les personnes qui possèdent ce nom ou ce prénom"
- "Je peux exporter mon carnet d'adresses au format HTML."

Les scénarios sont rédigés dans le langage du client, et décrivent des fonctionnalités dont l'implémentation paraît assez rapide – un scénario doit en effet pouvoir être complètement implémenté en une itération. Si un scénario paraît trop vague ou trop complexe, il est décomposé en scénarios plus simples. A l'inverse, des scénarios trop courts peuvent être regroupés en un seul pour obtenir la granularité souhaitée.

Au cours de la phase d'exploration, le client et l'équipe se forgent ainsi une idée assez précise du périmètre fonctionnel souhaité de l'outil. Ils établissent alors le premier plan de développement du projet.

➢ ***Planification du projet***

La planification est réalisée au cours d'une réunion dédiée, qui réunit l'équipe et le client et se déroule comme suit :

1. Le client présente les différents scénarios à l'équipe, qui tente de se faire une idée de la charge de travail de chacun d'entre eux.

2. L'équipe donne pour chaque scénario une estimation de son coût d'implémentation, en « points » abstraits : tel scénario coûte 3 points, tel autre 1 point, etc.

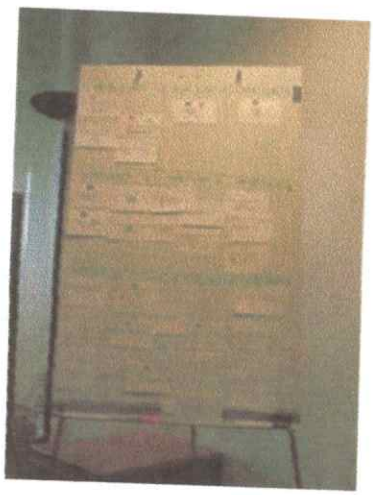
3. L'équipe donne au client une estimation de sa « vélocité », c'est-à-dire du nombre de points de scénarios qu'elle s'estime capable de traiter en une itération – par exemple 10 points.
   Au tout début du projet cette vélocité est seulement estimée, mais après les premières itérations elle est réajustée en adoptant la règle suivante : la vélocité estimée pour une itération donnée correspond exactement au nombre de points effectivement traités à l'itération précédente.

Remarque

La vélocité ne représente en aucun cas le niveau de "performance" de l'équipe, dans la mesure où elle dépend en grande partie de la façon dont les développeurs font leurs estimations. Cependant, ses variations peuvent indiquer des problèmes passagers : si la vélocité baisse brutalement, c'est peut-être que l'équipe a été ralentie pour une raison ou une autre, et il ne faut pas ignorer cette information.

Le client établit lui-même le plan de développement en affectant les différents scénarios aux itérations à venir, de sorte qu'à chaque itération la somme du nombre de points des scénarios choisis soit égale à la vélocité annoncée.

Côté outillage, on donne la priorité à l'aspect participatif de la démarche en notant les scénarios sur des fiches cartonnées que l'on colle sur un grand tableau ou un mur :



Cette pratique peut surprendre dans des contextes où les documents ont souvent une forme électronique, mais les séances de planification jouent avant tout sur l'aspect humain du projet, la communication directe entre l'équipe et le client permettant d'aligner les deux parties sur l'objectif à atteindre.
Le plan de développement ainsi constitué est disposé à proximité de l'endroit où travaille l'équipe, de manière à ce que celle-ci ait une vision toujours à jour de ses engagements.

➢ ***Première mise en production***

Comme nous l'avons évoqué plus haut, la phase d'exploration se termine par la première livraison du logiciel.
Cette première mise en production intervient aussi tôt que possible dans le projet : il faut trouver le contenu fonctionnel minimal qui commence à avoir un sens pour les utilisateurs, quitte à faire preuve d'imagination en amenant par exemple le nouveau logiciel en complément d'un système existant dans le cadre d'une fonctionnalité bien précise.

➢ ***Livraisons suivantes***

Les mises en production s'enchaînent ensuite à un rythme régulier, toujours fixé par le client. L'objectif est d'obtenir un feedback très rapide sur le développement – en pratique toutes les une à six itérations selon la complexité de déploiement du produit.
Le plan de développement est continuellement mis à jour si nécessaire pour tenir compte des événements suivants :

- L'équipe de développement progresse à une vitesse différente de celle prévue (dans le bon ou le mauvais sens)
- Le client décide de changer le contenu des itérations restantes. Il peut ainsi permuter des scénarios en fonction de nouvelles priorités, ou encore remplacer certains scénarios du plan par de nouveaux.

- ***Suivi du projet***

Le suivi « haut niveau » du projet peut par exemple s'appuyer sur des graphes inspirés de celui représenté ci-dessous, dans lequel on note le nombre de points de scénarios restant à implémenter.

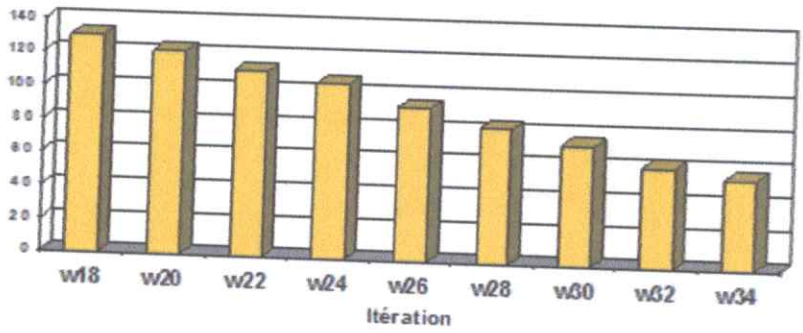


Figure A.2

Tous les scénarios imaginés par le client ne sont pas nécessairement représentés dans ce graphe. Seuls y figurent ceux qui ont été retenus pour la prochaine grande échéance – typiquement la prochaine version majeure du produit, ou encore la fin du projet pour un développement de type forfait.

Cette courbe permet par projection d'estimer si les objectifs du projet seront tenus, et elle permet aussi d'identifier certains « patterns » dans le fonctionnement de l'équipe - par exemple une tendance à la sur- ou sous-estimation des tâches (1).

- ***Tests de recette automatiques***

Pour chaque scénario planifié, un ensemble de tests de recette est écrit. Ces tests ont pour but de vérifier de manière automatique (c'est-à-dire sans intervention ou interprétation humaine) chacune des fonctionnalités demandées par le client. Le client définit ces tests et participe éventuellement à leur implémentation, assisté pour cela d'un certain nombre de testeurs.

Ces tests peuvent prendre plusieurs formes. Il peut s'agir de jeux de données, sous forme par exemple de feuilles de tableur ou de fichiers XML, qui définissent une transformation effectuée par le logiciel : « pour telle entrée, le logiciel doit produire tel résultat ». Il peut s'agir également de scripts pour les cas plus complexes, ces scripts

décrivant par exemple des séquences d'interactions de l'utilisateur avec l'interface graphique du produit.

Ces tests représentent dans un projet XP les spécifications détaillées de l'outil – des spécifications formelles, toujours en phase avec le développement. Dans un contexte « pur XP », ce sont les seules spécifications : il n'y a pas de *document* de spécifications à proprement parler. En pratique, cependant, des documents peuvent être exigés par l'organisation qui encadre le projet. Des documents synthétiques sont alors réalisés par le client.

- Les tests de recette sont lancés fréquemment, et tous les participants du projet (client, direction, développeurs) sont informés en permanence de leurs résultats (2).

• TRAVAIL EN EQUIPE

L'organisation d'une équipe XP s'éloigne de l'organisation traditionnelle d'une équipe de développement, dans lequel différents développeurs travaillent sur des parties distinctes de l'application, coordonnés par un chef de projet responsable de l'intégrité de l'ensemble. Ce schéma de *séparation* des activités fait place dans XP à un schéma d'*intégration*, ou de collaboration intensive.

➢ *Définition et partage des tâches*

Au début de chaque itération, le client, les développeurs et les testeurs se réunissent au cours d'une « réunion de planification d'itération ». La réunion commence par un point sur l'itération passée, puis le client décrit les scénarios à implémenter au cours de l'itération qui commence. S'ensuit une discussion sur les scénarios en question, au cours de laquelle l'équipe dresse une liste des tâches correspondantes.

Cette réunion se présente comme une réelle séance de conception collective, qui donne aux différents intervenants l'occasion d'aligner leur compréhension de ce qui doit être réalisé. Les besoins sont analysés à un niveau de détail plus fin, et les développeurs commencent à envisager les solutions techniques à mettre en place.

Au terme de cette réunion, l'équipe dispose d'une liste complète des tâches à réaliser dans l'itération. L'attribution des tâches se fait alors de manière auto organisée : les développeurs choisissent eux-mêmes leurs tâches en cours d'itération, ce qui est rendu possible par deux pratiques fondamentales d'XP : la responsabilité collective du code, et le travail en binômes.

- ***Responsabilité collective du code***

Chaque développeur d'une équipe XP est susceptible de travailler sur toutes les parties de l'application, sans être bloqué par une éventuelle spécialisation initiale. Cette liberté d'action s'accompagne toutefois d'une responsabilité : chaque membre de l'équipe est garant de la qualité de l'ensemble de l'application – en particulier, chacun a le devoir de laisser le code qu'il parcourt aussi clair et propre que possible, pour faciliter le travail de ceux qui y interviendront par la suite.

- ***Travail en binômes***

Les développeurs d'une équipe XP travaillent quasiment tout le temps en binômes, c'est-à-dire à deux sur un même poste. L'idée n'est en aucun cas d'avoir une personne qui code pendant que l'autre la surveille, mais au contraire d'aboutir à un dialogue permanent par lequel les deux membres du binôme sont engagés à 100% dans leur tâche.

Les binômes changent fréquemment (typiquement tous les jours), de sorte que chacun est rapidement amené à travailler avec tout le reste de l'équipe. C'est cela qui rend possible la pratique de responsabilité collective du code : un développeur donné peut intervenir sur une partie de l'application qu'il ne connaît pas encore bien, sachant qu'il pourra s'associer à un binôme ayant déjà travaillé sur le sujet. Après quelque temps de ce traitement, chacun a une bonne vision d'ensemble de l'application et peut intervenir n'importe où.

Cette mécanique de rotation des binômes et de responsabilité collective du code conduit à une très grande souplesse en matière d'organisation d'équipe : le projet

n'est plus bloqué par l'absence de tel ou tel individu, et toute l'équipe peut focaliser ses efforts sur une partie donnée de l'application en cas de besoin.

- ***Stand-up meetings***

Chaque jour, toute l'équipe se réunit pour un « stand-up meeting » d'une quinzaine de minutes - les participants se tiennent d'ailleurs tous debout (d'où le nom) pour éviter que cela ne traîne. A l'occasion de cette réunion, tous les membres de l'équipe prennent la parole pour faire un point sur l'avancement de l'itération, et surtout pour organiser la journée de travail à venir.

- ***Règles de codage et métaphore***

Des règles de codage sont définies et suivies par l'ensemble des développeurs. Elles donnent au code un aspect uniforme sur toute l'application, ce qui facilite le partage du code et le travail en binômes.

L'homogénéité du code ne se réduit toutefois pas au respect de quelques règles de programmation. L'équipe s'appuie en effet sur un système de métaphores commun, c'est à dire un ensemble d'expressions décrivant les acteurs du système et leurs interactions. Ce système de métaphores constitue :

- Un vocabulaire commun qui va permettre à toute l'équipe de parler des mêmes choses avec les mêmes mots. Par exemple, dans le monde des Télécom, il faut définir ce qu'est un "noeud" d'un réseau et se servir de ce mot dans le cadre de la définition établie.

- Une métaphore de fonctionnement, par exemple : "le logiciel de contrôle de cette machine-outil fonctionne comme une machine à café".

- ***Intégration continue***

L'intégration des modifications menées par les développeurs est faite tous les jours. Dès qu'un binôme finit une tâche, il met à jour la version d'intégration en s'assurant que tous les tests de non-régression de l'application passent à 100%. La version

"livrable" du logiciel évolue donc chaque jour, reflétant fidèlement l'état d'avancement des développements.

➢ ***Le rôle du coach***

Dans un contexte où l'équipe s'organise elle-même en définissant et en choisissant ses tâches, le rôle de chef de projet tel que nous le connaissons devient inadéquat. L'activité de coordination de l'équipe est prise en charge par un « coach », qui est davantage un facilitateur qu'un donneur d'ordres. Le but du coach est en effet de former l'équipe aux pratiques XP, et ensuite de travailler en permanence sur les pratiques de l'équipe pour améliorer son fonctionnement et l'amener à l'autonomie.

Les fonctions hiérarchiques et administratives du chef de projet font l'objet dans XP d'un rôle particulier : le « manager ». Le manager est le patron de l'équipe, il s'assure qu'elle dispose des moyens est nécessaire à son fonctionnement, fait l'interface avec le reste de l'organisation, et vérifie enfin que les résultats sont bien là.

➢ ***Rythme durable***

D'une part, une équipe fatiguée ne produit pas un bon travail. D'autre part, s'il y a trop de retard, c'est qu'il y a un problème de fond et il est illusoire de chercher à le masquer par davantage de travail. Pour ces raisons, l'équipe adopte la règle suivante : pas d'heures supplémentaires deux semaines de suite.

- **Programmation pilotée par les tests** :

La démarche extrêmement itérative proposée par XP n'est viable que si l'équipe est en mesure de garder la maîtrise technique de l'application, c'est-à-dire de faire en sorte que les modifications du code restent longtemps faciles et rapides malgré les nombreuses évolutions. Cette maîtrise n'est pas le fruit du hasard, elle est le résultat d'une discipline de développement qui s'appuie sur les pratiques suivantes :

➢ ***Mise en place intensive de tests unitaires***

En complément des tests de recette, qui servent à prouver au client que le logiciel remplit ses objectifs, XP utilise intensivement les tests unitaires de non-régression. Ces tests sont écrits par les développeurs en même temps que le code lui-même, pour spécifier et valider le comportement de chaque portion de code ajoutée.
Quasiment chaque classe de l'application possède une classe jumelle de test. Lorsque des développeurs doivent ajouter une nouvelle méthode à la classe applicative, ils commencent par ajouter à la classe de test un ensemble d'assertions qui décrivent le comportement de la méthode à ajouter. La méthode n'est implémentée qu'ensuite, pour que ces tests passent. Il est important de noter que les tests s'exécutent sans aucune interprétation de la part du développeur : les assertions en question doivent vérifier automatiquement si le code testé fonctionne ou non.

Les classes de tests ne sont pas jetées après usage : elles sont gérées par un framework dédié (1) , qui permet de les exécuter individuellement ou en totalité. La batterie de tests de l'application ne cesse donc de s'enrichir, et forme avec le temps un filet de sécurité qui permet aux développeurs d'effectuer des modifications dans le code existant sans crainte de régression.

Ces tests sont lancés à longueur de journée par les développeurs - en fait, à chaque compilation. Tous les tests unitaires doivent passer à 100% dans l'environnement d'un binôme avant tout report de modifications dans la version d'intégration du logiciel.

Cette pratique constitue l'une des pratiques centrales de XP. Ses avantages sont tels que chaque équipe, XP ou non, se doit de s'y pencher avec la plus grande attention :

- Les erreurs sont très vite détectées et localisées.
- Les tests donnent une vision précise de la tâche à réaliser et aident les développeurs à aller droit au but.

- Les tests placent les développeurs dans un contexte réduit qui leur permet de s'abstraire de la complexité du reste de l'application.

- Il est possible de s'assurer du fonctionnement global du système très rapidement, après toute modification du code, avant ou après toute intégration dans la version officielle du système.

Ce que l'on constate au final, c'est que les développeurs d'une équipe XP passent plus de temps à ajouter des fonctionnalités, et moins de temps à investiguer des défauts dans le code existant.

➢ *Remaniement du code*

L'application est toujours maintenue aussi malléable que possible grâce à une activité permanente de remaniement (*refactoring*). Le remaniement consiste à retoucher ou réorganiser des portions de code existantes, à comportement fonctionnel identique, en vue d'améliorer la structure d'ensemble de l'application.
L'une des motivations principales de remaniement est l'absence de duplication dans le code : tout doit y apparaître "Once and Only Once". Lorsqu'un développeur doit utiliser une partie de code qui se trouve déjà quelque part, il remanie l'application de manière à pouvoir utiliser directement le code existant. De la sorte les modifications de l'application n'impactent généralement que peu d'endroits dans le code, ce qui allège d'autant le travail.

Le remaniement est également utilisé pour rendre le code plus clair : lorsqu'un développeur intervient sur une partie donnée du code, il la modifie si nécessaire afin de faciliter le travail de ceux qui passeront après lui.
Le remaniement est plus qu'un moyen de dépoussiérer le code : c'est une façon de faire émerger une conception aussi adaptée que possible aux besoins de l'application, en supprimant au fur et à mesure tout ce qui nuit à sa simplicité et peut ralentir l'équipe. Comme les tests unitaires, le remaniement est pratiqué à longueur de journée - il s'agit d'une discipline quotidienne de qualité, et non d'une activité épisodique de réécriture.

➢ ***Conception simple***

"*You ain't gonna need it!*" : on ne fait de conception que pour les fonctionnalités existantes, pas pour les fonctionnalités futures. En corollaire :

- On ne fait de généralisations dans la conception que lorsque des besoins concrets se font sentir.

- On n'introduit pas d'optimisations si elles ne sont pas demandées par le client.

En somme, plutôt que de préparer le logiciel pour un futur hypothétique, XP préconise de s'assurer que le travail actuel soit toujours bien fait (code testé, simple, lisible et sans duplications) de sorte que les changements restent faciles et rapides le moment venu.

Contrairement à ce que l'on pourrait penser au premier abord, cette position n'est pas celle de développeurs désireux de se consacrer à l'implémentation sans perdre de temps à concevoir. Il s'agit plutôt de l'inverse : en s'interdisant d'avoir un code médiocre, une équipe XP fait de la conception une activité vitale pour son fonctionnement. On observe d'ailleurs en pratique que les développeurs d'une équipe XP passent la plupart de leur temps à travailler sur cette conception

LES VALEURS D'XP

Les processus agiles tendent à s'éloigner d'une approche Taylorienne, à la fois « scientifique » et impersonnelle, du travail. Ils mettent au contraire l'accent sur l'élément humain, et parient sur la performance d'une équipe dont les membres travaillent en collaboration étroite – l'équipe en question incluant notamment la maîtrise d'ouvrage du projet.

Plus généralement, les pratiques XP sont sous-tendues par les quatre valeurs suivantes :

- **Communication** : XP favorise le contact humain, la communication directe, plutôt que le cloisonnement des activités et les échanges de courriers électroniques ou de documents formels. Les développeurs travaillent directement avec la maîtrise d'ouvrage, les testeurs sont intégrés à l'équipe de développement, etc.

- **Feedback** : qu'il s'agisse d'itérations courtes, de livraisons fréquentes, de travail en binômes ou de tests automatiques exécutés en permanence, la plupart des pratiques XP sont conçues pour donner un maximum de feedback sur le déroulement du projet afin de corriger la trajectoire au plus tôt. En particulier, les points de début d'itération offrent à l'équipe le moyen de prendre du recul sur son fonctionnement et de l'améliorer sans cesse au fil des itérations.

- **Simplicité** : comme nous l'indiquions au début de ce dossier, XP relève le défi suivant : « que pouvons-nous arrêter de faire tout en continuant à créer efficacement un logiciel qui réponde aux besoins réels du client ? ». Cette recherche de simplification touche le processus lui-même, mais aussi l'outil fabriqué (la mécanique de planification incite le client à focaliser les efforts sur les fonctions prioritaires) ou encore la conception de l'application (guidée par un principe de « You ain't gonna need it »).

- **Courage** : il s'agit principalement du courage d'honorer les autres valeurs – celui de maintenir une communication franche et ouverte, d'accepter et de traiter de front les mauvaises nouvelles, etc.

C'est en définitive sur ces critères qu'une équipe XP juge de sa maturité et trouve les principaux axes de son amélioration

Annexe C :

La bibliothèque OpenGL

(Open Graphic Library)

C

INTRODUCTION

Aujourd'hui, lorsqu'on parle de bibliothèque de développement 3D, les mots qui reviennent le plus souvent sont **Direct3D** et **OpenGL.** Direct3D est une API (Application Programming Interface) développée par Microsoft pour son OS Windows. Propriétaire et non portable, Direct3D fait partie de l'ensemble DirectX et est relativement orienté 'Jeux vidéo'. Son principal avantage est d'être bien supporté par les constructeurs de cartes graphiques grand public.

Pour sa part, **OpenGL** (Open Graphic Library, ce que se traduit en bon français par 'Bibliothèque Graphique Ouverte') est une API développée depuis 1992 par Silicon Graphics , un des grands ténors de l'informatique graphique professionnelle. Devenue aujourd'hui un standard industriel, elle a été implémentée sur la plupart des plates-formes informatiques actuelles (Linux et autres Unix, Windows, MacOs, BeOS...) et est disponible pour de nombreux langages de programmation (C, C++,Java,Fortran,Ada...).

OpenGL en action dans Blender

Originalement développée pour les stations graphiques haut de gamme Silicon Graphics, l'aspect ouvert d'**OpenGL** lui a valu d'être porté sur la plupart des plateformes modernes, et sur la plupart des systèmes d'exploitation. Aujourd'hui, OpenGL est reconnu par les professionnels comme un standard, et est utilisé par :

Aujourd'hui de nombreuses applications renommées utilisent OpenGL, dans différents domaines :

- **Modélisation et animation 3D** : 3DSMax, Maya, Softimage, Blender...
- **C.A.O.** : Catia, Pro-Engineer, SolidWorks...
- **Jeux Videos** : Quake (1,2 et 3), Half-Life, Unreal...

FONCTIONNALITES D'OPENGL

La première question qui vient évidemment à l'esprit est la suivante : **Que peut on faire avec OpenGL ?** La réponse est la suivante : OpenGL permet de représenter à l'écran des scènes tridimensionnelles réalistes. Le terme 'réaliste' est ici tout a fait relatif : on peut considérer comme réaliste une image de synthèse ou tous les objets représentés sont correctement placés au niveau de la perspective sans que les couleurs correspondent à ce qu'on observe dans la nature. Le réalisme ultime peut être atteint lorsqu'il est impossible à première vue de savoir si une image est issue d'un calcul ou si elle simplement une photo numérisée (on parle alors de photoréalisme). Au risque d'en décevoir certains, OpenGL ne vous permettra pas de générer des images photoréalistes.

OpenGL est résolument orienté vers l'interaction 'temps réel', c'est à dire qu'un programme conçu avec OpenGL permet en général à l'utilisateur d'agir par l'intermédiaires de périphériques d'entrée (clavier, souris, trackball...) sur la scène (déplacement de caméra, changement de focale, déplacement des objets....), et d'en visualiser immédiatement l'effet. Ainsi, lorsque vous jouez à Quake, chaque fois que

vous déplacez le joueur (on considère que la caméra est placée dans l'oeil du joueur), le programme recalcule la nouvelle image et l'affiche à l'écran. On comprend aisément que pour que le jeu vidéo soit 'fluide', c'est-à-dire que les déplacements du joueur ne paraissent pas trop saccadés, le programme doit être capable de calculer et d'afficher une image extrêmement rapidement (moins d'un dixième de secondes).

Dans le cas d'un jeu vidéo on peut considérer que si l'image n'est pas recalculée une quinzaine de fois par secondes, le jeu ne sera pas fluide.

Bien que non orienté photoréalisme, OpenGL permettent d'obtenir des rendus tout à fait satisfaisants. Voici une liste non exhaustive des principales fonctionnalités d'OpenGL.

- Affichage de maillages polygonaux.
- Rendu par projection orthogonale ou perspective.
- Placage de texture.
- Simulation d'éclairages réalistes
- Ombre portées

- Insertion d'objets bitmaps en avant plan (images, polices)
- Effet de brouillard, de flou
- Gestion des courbes et surfaces gauches (Splines, Nurbs...)
- ...

GLUT

Pour créer un programme OpenGL dans un environnement fenêtré comme X-Window, il faut être en mesure de créer une fenêtre pour y afficher nos images, de la détruire, et de gérer les interactions avec l'utilisateur par le clavier, la souris et tous les autres types de périphériques d'entrée. OpenGL se voulant indépendant de toute plate-forme matérielle, l'API ne fournit pas de telles fonctions. Heureusement, il

existe une bibliothèque annexe, nommée **Glut** (GL Utility Toolkit) qui fournit ces fonctions. Elle est développée par Mark Kilgard . Glut est fourni avec les dernières

moutures de Mesa. Cependant, certaines versions de Mesa fournies avec les distributions ne comportent pas glut.

Voici une liste de toutes les fonctionnalités proposées par Glut :

- Gestion de fenêtres.
- Gestion des événements par fonctions de rappel.
- Gestion de périphériques d'entrée exotiques (spaceballs...).
- Gestion des polices de caractères.
- Fonctions de création de menus