

ACADÉMIE DE MONTPELLIER
UNIVERSITÉ MONTPELLIER II
— SCIENCES ET TECHNIQUES DU LANGUEDOC —

MÉMOIRE DE STAGE DE MASTER

SPÉCIALITÉ : **Recherche en Informatique**
Mention : **Informatique, Mathématiques, Statistiques**

effectué au EUROCONTROL Experimental Center

—
sous la direction de PR. VU DUONG

Rapport de Stage de DEA

par

Loic Baud

Soutenue le 19 juin 2006

Résumé

Dans ce rapport de Master touchant au domaine de l'haptique, il sera présenté, dans une première partie, au lecteur une définition de l'haptique, un état de l'art globale de cette discipline ainsi qu'une description plus détaillée des différents modèles de rendu haptique pour les environnements composés d'objets non déformables, à trois et à six de degrés de liberté.

Il sera aussi exposé dans ce document un nouveau modèle de rendu haptique pour les environnements non déformables à six degrés de liberté et un nouvel algorithme de détection d'intersection entre un segment et un polygone.

Les implémentations et les performances de ce nouveau modèle et de ce nouvel algorithme seront également étudiés.

Dans une seconde partie, une courte introduction sur l'algèbre géométrique sera présentée, révélant une partie du grand potentiel de cette théorie.

Abstract

In this report of Master dealing with haptic, it will be shown, in a first part, a definition of haptic, a global state of the art of this discipline and also a more detailed description of the different models of haptic rendering of three and six degrees of freedom environments.

It will also be exposed in this document a new model of haptic rendering for the solid-composed six degree of freedom environment and a new Ray-Polygon Intersection algorithm.

Their implementations and performances will also be studied.

In a second part, a short introduction of geometric algebra will be presented, showing a part of the great potential of this theory.

Table des matières

1	INTRODUCTION	4
1.1	Définitions et notions d'haptique	4
1.1.1	Préambule	4
1.1.2	L'haptique	5
1.1.3	Historique	5
1.1.4	Le rendu haptique	6
1.2	Domaines de recherche en haptique	7
1.2.1	Détection de collision	7
1.2.2	Qualité du rendu haptique	8
1.2.3	Perception des textures	8
2	ETAT DE L'ART	10
2.1	Environnement à 3DOF	10
2.1.1	Le modèle GOD-Object	10
2.1.2	Le modèle basé sur le proxy virtuel	11
2.2	Environnement à 6DOF	12
2.2.1	Le modèle basé sur les LODs	12
2.2.2	le modèle LMD	15
2.2.3	le modèle VoxMap-PointShell	17
3	UN MODELE HAPTIQUE BASE SUR LA DETECTION DE COLLISION SEGMENT-POLYGONE	20
3.1	Introduction	20
3.2	Principe	20
3.2.1	La détection de collision	21
3.2.2	Génération des forces et des couples de réponse	22
3.2.3	Analyse des avantages et des défauts de ce modèle	23
3.3	Implémentation	26
3.4	Un nouvel algorithme pour la détection d'intersection segment- polygone	28
3.4.1	Introduction	28
3.4.2	L'algorithme de Badouel	29
3.4.3	L'algorithme de Möller-Trumbore	30
3.4.4	Algorithme du sens de rotation	30

3.4.5	Comparaison du coût de ces algorithmes	32
3.4.6	Vérification expérimentale	33
3.5	Performances	34
4	PETIT DETOUR PAR L'ALGÈBRE GEOMETRIQUE	36
4.1	Préambule	36
4.2	Histoire de l'algèbre géométrique	36
4.3	Introduction mathématique à la géométrie algébrique	37
4.3.1	Espaces vectoriels	37
4.3.2	Le produit intérieur (Inner product)	37
4.3.3	Le produit extérieur (Outer product)	38
4.3.4	Le produit géométrique	39
4.4	Algèbre géométrique dans \mathbb{R}^3	39
4.4.1	Algèbre géométrique dans un espace à trois dimensions \mathcal{G}^3	40
4.4.2	Transformations sur les éléments de \mathcal{G}^3	40
4.5	L'algèbre géométrique conforme (CGA)	42
4.5.1	Passage à l'espace conforme	42
4.5.2	Transformations	43
4.5.3	Les objets primitifs dans la CGA	44
5	CONCLUSION	46
5.1	Travaux accomplis	46
5.2	Travaux éventuels à faire	47
5.3	Impressions sur ce stage	47

Chapitre 1

INTRODUCTION

1.1 Définitions et notions d'haptique

1.1.1 Préambule

Ce stage a été effectué au sein du département INO du centre expérimental d'eurocontrol (EEC) à Brétigny sur Orge. Eurocontrol est l'agence européenne pour l'organisation et la sécurité du trafic aérien, le principal but de l'agence, qui compte 36 membres, est d'améliorer la gestion du trafic aérien (ATM) européen pour faire face à l'évolution grandissante de celui-ci. Elle cherche donc à maintenir un très haut niveau de sécurité et à réduire les coûts des vols tout en respectant l'environnement.

La mission du centre expérimental est de jouer un rôle de premier plan dans la recherche pour l'amélioration de l'organisation du système actuel de gestion du trafic aérien.

Les recherches menées dans le centre expérimental se décomposent en 5 domaines :

- Gestion de l'aéroport.
- Gestion des capacités et demandes du réseau.
- Secteur sécurité et productivité.
- Secteur Société-Environnement-Economie.
- Recherche innovante (INO).

Il m'a été demandé lors de ce stage dans le cadre d'un projet visant à remplacer les écrans radar de contrôle classique par un système composé d'éléments de vision en trois dimensions et d'éléments de contrôle haptique, de construire un moteur haptique gérant les environnements composés d'objets solides (i.e. objets non déformables). Il ne sera donc pas, dans ce rapport, question d'étudier le domaine de l'haptique de l'environnement avec objets déformables.

1.1.2 L'haptique

L'haptique est l'étude du sens du toucher et de l'interaction avec un environnement via le sens du toucher, un dispositif haptique est donc un dispositif qui permet l'utilisation du sens du toucher pour interagir avec un environnement. On peut décomposer le sens du toucher en trois sens différents, le sens tactile, le sens thermique et le sens kinesthésique.

Le sens tactile correspond à la sensation des textures, c'est celui là qui permet de différencier les différents types de surface des objets, par exemple c'est ce sens qui nous permet de décider si l'objet est lisse ou bien rugueux.

Le sens thermique, comme son nom l'indique, nous donne des renseignements sur la chaleur des objets des objets, mais aussi sur leur nature, ce sens nous permet d'affirmer que le bois est plus "chaud" que le métal (bien qu'étant en réalité tout les deux à la même température) et donc de différencier ces deux matériaux.

Le dernier sens, le sens kinesthésique, nous renseigne sur les forces s'exerçant sur notre corps, par exemple lorsque nous serrons un objet dans la main, nous exerçons des forces sur celui-ci et par le principe d'action-réaction l'objet exerce aussi des forces sur notre main, c'est le sens kinesthésique qui nous renseigne sur ces forces. Bien que la technologie actuelle permette d'imaginer des dispositifs haptiques utilisant les deux premiers sens, c'est essentiellement le troisième qui est utilisé et ce sont les dispositifs haptiques basés sur celui ci dont il sera question dans ce document.

Il a été montré par Insko en 2001 [1] que le sens du toucher augmente l'impression d'immersion dans le monde virtuel, de même que les retours de forces augmentent les performances dans des applications telles que l'assemblage d'objets virtuels. Les dispositifs haptiques peuvent donc se montrer particulièrement utiles en ce qui concerne l'entraînement pour des tâches hautement dangereuses, telles que les simulations de pilotages ou bien les entraînements de chirurgie des jeunes médecins. Les dispositifs à retour de force permettent aussi d'acquérir plus facilement des informations par le sens du toucher, il pourrait donc s'avérer utile dans un environnement surchargés d'informations tel que l'on peut en trouver dans les différents domaines du contrôle aérien.

1.1.3 Historique

La première apparition des dispositifs haptiques se fit dans le livre de fiction "Brave New World" par Aldous Huxley, celui-ci décrit théâtre où les bras des sièges étaient munis d'un dispositif haptique, le premier dispositif haptique réel remonte quant à lui à 1952 où un système maître-esclave a été établi par Goertz au laboratoire national d'Argonne, Etats-Unis, pour manipuler à distance les substances radioactives, ce système se composait d'un servomécanisme électrique qui recevait le signal de retour des capteurs de la partie esclave et générer les forces de retour.

Ce genre de système était basé sur de la mécanique et les forces de retour étaient non pas calculé mais "ressentie" par des capteurs, le premier grand pas

en avant vers la conception actuelle de l'haptique fut l'idée de remplacer la partie mécanique esclave par une simulation mathématique qui calculerait les forces de retour, le projet GROPE de l'université de Chapel Hill (Caroline du nord) conduit par Brooks, Jr. et al. qui a débuta en 1967 et se termina en 1990 [2] fut le premier à le faire ¹.

Le second pas de géant en avant fût la séparation de la configuration cinématique de la partie maîtresse et de la partie esclave, elle fut mise en oeuvre la première fois par Bejczy et Salisbury en 1980 [3], dans le cadre de contrôle de système de télémanipulateurs.

Ces premiers systèmes haptiques étaient tout juste capable de simuler des interactions avec de très simples objets virtuels, le premier à pouvoir interagir avec un environnement géométrique complexe fut probablement le *sandpaper* de Minsky et al. en 1990 [4], il était suffisamment performant pour l'exploration des objets texturés. Quelques années après, en 1994, Massie et Salisbury [5] développèrent le PHANToM une interface haptique de la forme d'un bras articulé avec un stylet à son extrémité, c'est encore de nos jours, sous une version plus évoluée, le dispositif le plus courant et le plus commercialisé, c'est également un modèle de PHANToM qui sera utilisé lors de ce stage.

1.1.4 Le rendu haptique

Le rendu haptique est défini tel qu'étant le processus qui calcule et génère les forces en réponses aux interactions des utilisateurs avec les objets virtuels.

Il y a différents types d'algorithmes de rendu haptique plus ou moins complexes suivant les degrés de liberté dans l'environnement virtuel dans lequel nous évoluons, si il ne s'agit que de toucher les objets virtuels avec un point, nous évoluons alors dans un environnement à trois degrés de liberté (3DOF, 3 degree of freedom). La plupart des travaux existant portent sur le rendu haptique dans un environnement à 3DOF, le problème se pose de la façon suivante, étant donné un objet virtuel O et la position d'un point virtuel p gouverné par l'interface haptique il s'agit alors de savoir si le point p est en contact avec la surface de l'objet O . En considérant la surface de O comme étant une collection de n triangles, trouver si un point p est en contact avec l'objet O a dans le pire des cas une complexité en $O(n)$, en utilisant le partitionnement de l'espace et des prédictions de mouvement Gregory et al. a démontré en 1999 [6] qu'il est possible dans de nombreux cas de ramener la complexité de cette opération à $O(1)$.

Le rendu haptique dans un environnement à 3DOF avec un coût si faible propose des solutions intéressantes pour beaucoup d'applications avec un besoin de retour de force, telle que la peinture, la sculpture virtuelle et l'entraînement à diverses opérations chirurgicales. Le fait que l'interaction avec l'environnement de ces applications soit suffisamment petit pour être assimilé à un point, et donc être traitée par un rendu haptique 3DOF.

¹Ce projet avait pour but de calculer les interactions et les forces d'attraction entre diverses molécules

Cependant dans la vie de tous les jours, lorsque que nous nous brossons les dents ou que nous utilisons un stylo, nous faisons interagir des objets avec d'autres objets, et nous ressentons les forces à travers l'objet que nous manipulons. Nous évoluons alors dans un environnement à 6 degrés de liberté (6DOF), nous pouvons faire subir des translations et rotations aux objets que nous manipulons, des algorithmes de rendu haptique existe aussi pour ces environnements à six degrés de liberté. Dans de tels environnements, il ne s'agit plus de calculer seulement des forces, mais aussi les couples associés aux rotations des objets manipulés lors de collisions avec un objet virtuel, de plus lorsqu'un utilisateur manipule un objet et interagit avec l'environnement, cette interaction ne se limite plus à un seul contact point contre surface, on a bien souvent une multitude de point de contact. La complexité de rendu haptique dans ces environnements est largement supérieure à celle du rendu haptique dans un environnement à 3DOF.

Les techniques existantes de rendu haptique des environnements à 6DOF peuvent se décliner selon deux catégories, la première est le *direct rendering* (le rendu direct), ces techniques calcul la position et l'orientation de l'objet manipulé par l'utilisateur et les appliquent directement à celui-ci. La détection de collision s'effectue entre l'objet manipulé et les objets virtuels elle est calculé en fonction de la pénétration de l'objet dans les objets virtuels, les forces et couples de retour sont en suite calculés et directement renvoyé à l'utilisateur

La seconde catégorie de technique se nomme le *virtual coupling*, cette technique calcule la position et l'orientation de l'objet manipulé mais contrairement au *direct rendering* il ne l'applique pas directement à l'objet. Elle l'applique à un objet virtuel qui est couplé à l'objet contrôlé par l'utilisateur par un lien viscoélastique. Les forces appliquées à l'objet contrôlé à l'utilisateur sont donc les résultantes de la liaison viscoelastique.

1.2 Domaines de recherche en haptique

1.2.1 Détection de collision

La détection de collision entre divers objets géométriques n'est pas uniquement un problème rencontré en haptique, ils se retrouvent dans de très nombreuses disciplines mathématiques et informatiques, c'est donc un problème récurrent et assez bien documenté. Il existe plusieurs représentation des objets géométriques, soupe de polygones, surfaces courbes, algèbre géométrique et les modèles polyédriques et pour chacune de ces représentations plusieurs méthodes pour détecter les collisions. Sans plus rentrer dans les détails des différentes représentations et des différents algorithmes, les détections de collisions entre un objet et un environnement se font en général en deux étapes, tout d'abord on élimine tous les objets de l'environnement qui sont suffisamment géographique-ment éloignés de l'objet traité pour qu'il n'y ait pas de collision possible par un partitionnement de l'espace en général, par la suite les algorithmes détectent si il y a une collision entre les objets de l'environnement restant et l'objet traité,

cette détection de collisions est un point critique du rendu haptique parce que c'est à partir des caractéristiques des éventuelles collisions que seront générés les forces de retour, et ces collisions détectées doivent donc être le plus précises possible afin que les forces de retour générées soient les plus précises possibles.

1.2.2 Qualité du rendu haptique

Nous pouvons juger la qualité d'un rendu haptique par deux critères qui sont la vitesse de réaction et la qualité de celle-ci.

La vitesse de réaction peut se décrire de la façon suivante, dans un espace sans contact l'objet manipulé par l'utilisateur doit répondre immédiatement au mouvement de celui-ci, par contre si l'objet rentre en contact avec un élément de l'environnement, l'objet manipulé doit immédiatement s'arrêter et l'utilisateur immédiatement ressentir ce contact.

La qualité de la réaction quant à elle est fonction de la différence des forces que l'utilisateur ressent en l'absence de contact entre l'objet qu'il manipule et l'environnement et de celles qu'ils ressent si il y a un contact. Idéalement il ne devrait pas y avoir de force lorsque l'utilisateur bouge l'objet dans un espace sans contact et elle devrait être proportionnelle à celle que l'utilisateur produit sur le dispositif haptique en cas de contact.

La qualité du rendu influe évidemment sur la perception de l'utilisateur, et des phénomènes de latence du à une mauvaise vitesse de réaction sont perçues par l'utilisateur comme de petites oscillations, suffisamment nuisibles pour que l'utilisateur soit déstabilisé, une solution pour augmenter cette qualité est d'avoir un haut taux de rafraîchissement des forces. Plusieurs expériences, notamment celle de Colgate et Brown en 1994 [7] et celle de Brooks, Jr. et al. en 1990 [2], montre que les utilisateurs sont capables de ressentir la différence de qualité sur des plage de fréquence de taux de rafraîchissement allant 500Hz à 1000Hz. De ces expériences nous pouvons déduire que si nous voulons avoir une qualité de rendu optimale, un des moyens est d'avoir un taux de rafraîchissement supérieur à 1kHz.

1.2.3 Perception des textures

Une surface texturée fut définie par Klatzky et Lederman en 2003 [8] comme étant une surface homogène avec de très légères irrégularités , il existe deux branches de recherche différentes sur la perception des textures, la perception direct d'une texture avec la peau (environnement à 3 DOF) et la perception de texture via un objet intermédiaire (environnement à 6 DOF).

La plupart des recherches sur le sujet de la perception des textures se sont concentrés sur la première branche, la perception cutanée. Les premières recherches sur le sujet avait aboutit à la conclusion suivante, la perception de la rugosité se faisait par l'intermédiaire des vibrations engendrées par le contact direct de la peau sur la surface. En 1992, Connor et Johnson [9] démontrèrent pour leur part distribution des zones de pression joue est responsable de la reconnaissance des textures larges, c'est à dire avec des aspérités plus grandes que

1 mm.

L'autre partie concernant la perception des textures des objets par le contact avec d'autres objets a été quand à elle longtemps ignoré. Ce n'est qu'en 2002 que Klatzky et Lederman ont mis en évidence que la perception des textures par le contact avec d'autres objets est de nature vibratoire [10].

Les expériences conduites par Lederman et al. [11] [12] pour caractériser la perception des textures, consistaient à faire explorer par un utilisateur une surface texturée à l'aide d'un objet sphérique, l'utilisateur devait ensuite donner un avis subjectif sur la texture rencontrée. De ces études Lederman et al. démontrèrent que la vitesse d'exploration ainsi que les forces appliquées par l'utilisateur avait une importance singulière.

En 2004, Otaduy et al. [13] intégrèrent toutes ces notions dans un moteur de rendu haptique, c'est à ce jour le modèle haptique de perception des textures le plus élaboré existant.

Chapitre 2

ETAT DE L'ART

Dans cette partie seront présentés les principaux et les plus performant modèles de moteur haptique, pour les environnements à 3DOF et à 6DOF.

2.1 Environnement à 3DOF

2.1.1 Le modèle GOD-Object

Ce modèle proposé par Zilles et Salisbury (dans l'article [14]), utilise un point virtuel appelé le GOD-Object afin de calculer les forces de retour et la position du point contrôlé par l'utilisateur. Ce point, le GOD-Object (GO) est calculé à l'aide de surfaces jouant le rôle de contraintes locales et de la position du point contrôlé par l'utilisateur avant la gestion des collisions.

La position du GO est définie comme suivant :

- si le point contrôlé par l'utilisateur n'est pas à l'intérieur d'un objet alors la position du GO est égale à la position de celle du point contrôlé.
- si le point est à l'intérieur d'un objet de l'environnement alors la position du GO est la position du point le plus proche de la position du point contrôlé par l'utilisateur qui respecte les contraintes.

Pour calculer ce point il faut donc minimiser le système suivant, soit :

$$A_i \cdot x + B_i \cdot y + C_i \cdot z + D_i = 0$$

l'équation de la surface jouant le rôle de contrainte et UP la position du point de l'utilisateur, maintenant posons :

$$E = \frac{1}{2} \cdot k \cdot ((x - x_{UP})^2 + (y - y_{UP})^2 + (z - z_{UP})^2)$$

comme étant la fonction d'énergie associée à la distance du point contrôlé par l'utilisateur et le GO, alors la fonction de coût à minimiser peut-être définie en utilisant les multiplieurs de Lagrange comme ci dessous :

- GOD - Object
- Point contrôlé par l'utilisateur avant gestion de collision

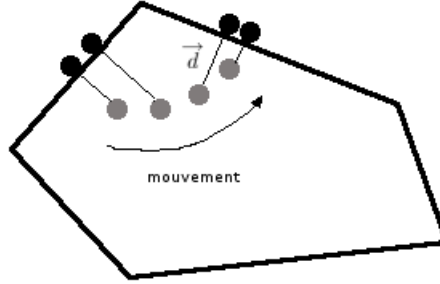


FIG. 2.1 – Le point contrôlé par l'utilisateur et son objectif, le God-Object

$$C = \frac{1}{2}.k.((x-x_{UP})^2+(y-y_{UP})^2+(z-z_{UP})^2) + \sum_{i=1}^n \lambda_i(A_i.x+B_i.y+C_i.z+D_i)$$

où n est le nombre de surface. Il faut donc trouver les valeurs de $x,y,z,\lambda_i \forall i$ qui minimisent la fonction de coût C .

Il a été démontré que pour $n \geq 4$ le système n'est plus solvable dans un temps raisonnable il est donc inutile de fonctionner avec plus de 3 surfaces jouant le rôle de contrainte.

Pour $n = 3$, par dérivation partielle sur $x,y,z, \lambda_1, \lambda_2, \lambda_3$ on obtient ces 6 équations linéaires suivantes :

$$\begin{bmatrix} 1 & 0 & 0 & A_1 & A_2 & A_3 \\ 0 & 1 & 0 & B_1 & B_2 & B_3 \\ 0 & 0 & 1 & C_1 & C_2 & C_3 \\ A_1 & B_1 & C_1 & 0 & 0 & 0 \\ A_2 & B_2 & C_2 & 0 & 0 & 0 \\ A_3 & B_3 & C_3 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} x_{UP} \\ y_{UP} \\ z_{UP} \\ D_1 \\ D_2 \\ D_3 \end{bmatrix}$$

Les résoudre et trouver les valeurs de $x,y,z, \lambda_1, \lambda_2, \lambda_3$ est alors très simple et peu coûteux en temps de calcul.

Une fois la position du GOD-Object trouvé la force de retour est calculé en fonction de la distance \vec{d} entre le GO et le point contrôlé par l'utilisateur.

2.1.2 Le modèle basé sur le proxy virtuel

Développé par Ruspini et al. (articles [15] et [16]) ce modèle est en fait une extension du modèle GOD-Object, la majeure différence est que les surfaces jouant le rôle de contraintes sont remplacées par la notion de C-obstacles inspirée du domaine de la robotique. Le GOD-Object est lui aussi remplacé, par le proxy

virtuel, qui a la possibilité contrairement à GO d'être relocalisé afin d'effectuer de permettre des calculs plus exacts.

Le calcul de la position se fait de façon itérative, et de la façon suivante :

- On procède au calcul de position du proxy virtuel, celle-ci étant égale à l'intersection du chemin parcourue par le point contrôlé par l'utilisateur et le premier C-obstacles rencontré.
On définit ainsi la position temporaire du proxy virtuel ainsi que du plan qui va servir de contrainte.
- Ensuite on calcule une nouvelle position temporaire pour le proxy virtuel en utilisant la même méthode de minimisation de distance que pour le GOD-Object, en utilisant la position et le plan calculé à l'étape précédente.
- On recommence les deux étapes précédentes jusqu'à ce que la position du proxy virtuel soit stable.

Un exemple illustré de la méthode de calcul de la position de calcul se trouve exposé dans la figure 2.2

2.2 Environnement à 6DOF

De nos jours la majorité des travaux en haptique porte sur ce domaine, et de ces recherches émanent trois principaux modèles haptiques le modèle basé sur les LODs de Otaduy et Lin, le modèle des distances minimums locales de Jonhson et Willemsen et le modèle VoxMap-Pointshell de McNeely, Puterbaugh et Troy. Dans cette partie seront exposé un résumé des articles décrivant et nécessaires à la compréhension de ces modèles.

2.2.1 Le modèle basé sur les LODs

Les LODs (niveaux de détails) sont déjà employés dans le domaines de l'affichage de scènes en 3D, dans l'article [17] est proposé un algorithme pour définir les LODs d'un objet géométrique défini par l'ensemble des triangles composant sa surface, ainsi qu'une applications de ces LODs dans le domaine haptique, plus précisément une application à la détection de collision.

Constructions des LODs

Les LODs, sont différents niveaux de résolutions d'un objet, prenons pour exemple un objet dont la surface est composée de 500 triangles, appelons ce niveau LOD_0 de résolution r_0 , alors le niveau de résolution suivant $r_0 + 1$ du LOD_1 aura sa surface composé de seulement 250 triangles et ainsi de suite jusqu'au niveau de résolution le plus bas choisi, on obtient alors une hiérarchie de LOD pour chaque objet.

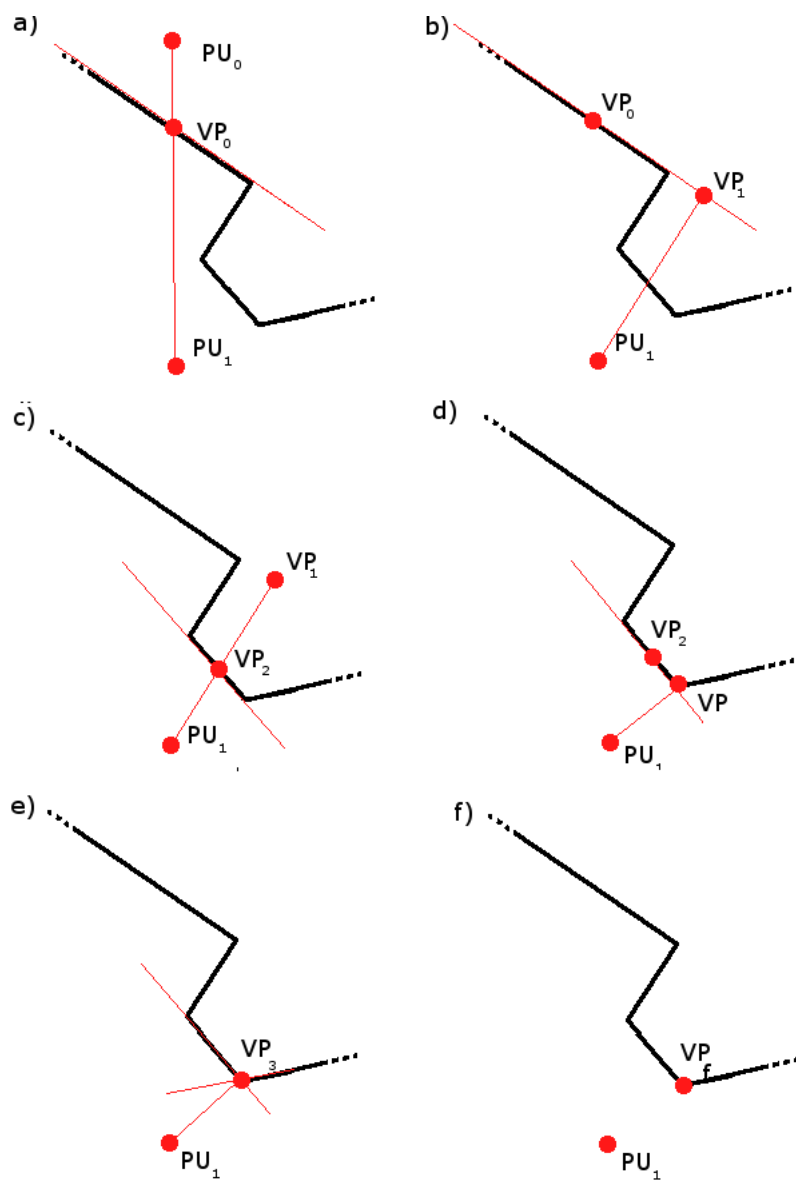


FIG. 2.2 – Exemple d'une étape de calcul et de positionnement du proxy virtuel.

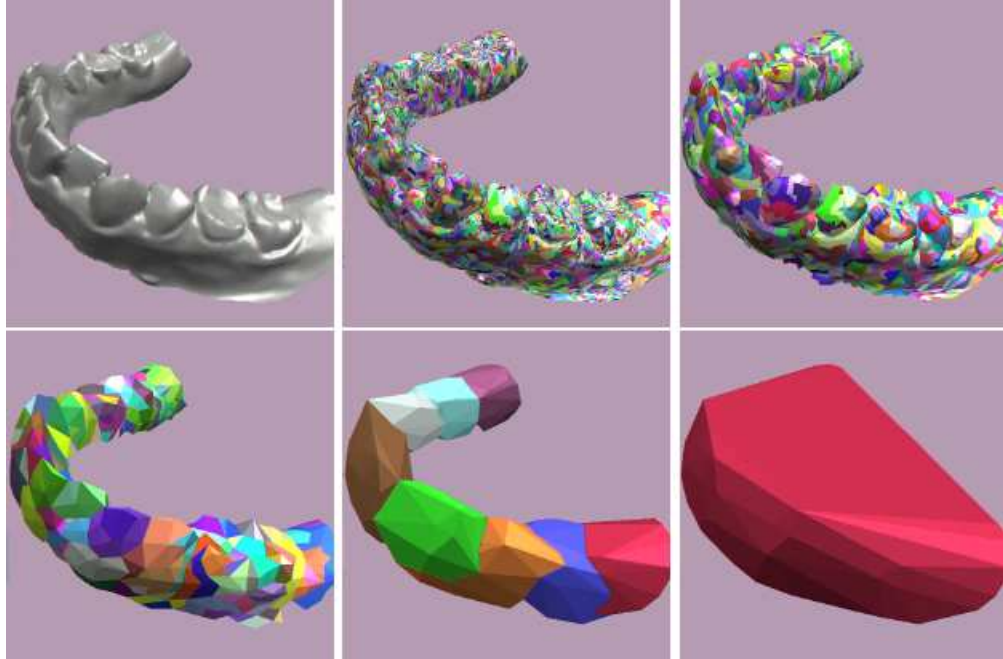


FIG. 2.3 – Les niveaux de détails successifs d’une machoire en 3D, de gauche à droite et de haut en bas résolution r_0 , r_3 , r_6 , r_{11} et r_{14}

Le principe de l’algorithme de génération de LOD utilisé dans cet article se fonde sur la technique de l’effondrement de cotés (edge collapsing) dont une des descriptions se trouve dans le rapport technique [18], le pseudo code de cet algorithme pour la création de cette hiérarchie de niveau de détail se trouve présenté figure 2.4.

Pour faire simple, cet algorithme se contente de décomposer l’objet de niveau de résolution r_0 en un ensemble de parties convexes, puis avec la technique de l’effondrement de cotés diminue le nombre de triangles de ces parties.

Puis l’algorithme recalcule la décomposition en parties convexes du nouvel objet obtenue, ceci nous donne le niveau de résolution $r_0 + 1$. L’algorithme se poursuit ainsi jusqu’au niveau de résolution souhaité.

Détection de collision

Une fois cette hiérarchie construite, nous obtenons pour chacun des objets un arbre de hiérarchies de volume englobant (voir la figure 2.5 a)). Puis de ces arbres nous déduisons, pour chaque paire d’objet que dont nous voulons savoir si ils sont en collision ou pas, un arbre (Bounding Volume Test Tree, figure 2.5 b)) qui vont servir à la détection des collisions. Cette détection se fait de la façon suivante, on commence par tester si il y a collision entre les objets

```

Compute surface convex decomposition
Dump initial LOD
n = number of convex pieces
Compute resolution of edges
Initialize edges as valid
Create priority queue
while Valid(Top(queue)),
  if FilteredEdgeCollapse(Top(queue)) then
    PopTop(queue)
    Recompute resolution of affected edges
    Reset affected edges as valid
    Update priority of affected edges
    Attempt merge of convex pieces
  else
    Set Top(queue) as invalid
    Update priority of Top(queue)
  endif
  if Number of pieces  $\leq n/2$  then
    Dump new LOD
    n = number of convex pieces
  endif
endwhile
while Number of pieces > 1,
  Binary merge of pieces
endwhile

```

FIG. 2.4 – Algorithme en pseudo code pour générer les LODs d'un objet

représenté dans leur résolution la plus basse, la racine de l'arbre (le noeud ab sur a figure 2.5 b)), cette collision est facile à détecter car à cette résolution la représentation des objets est très simple. Puis si il y a collision, on teste si il y a collision entre les fils de résolution supérieure. Et ainsi de suite jusqu'à ce qu'il n'y ait plus de collisions détectées ou que l'on ait atteint le niveau de résolution initial (collision détectée sur une feuille du BVTT). Dans le second cas cela signifie qu'il y a donc une collision dont la profondeur de pénétration est connue. On peut donc générer les forces de retour associées à la collision détectée.

2.2.2 le modèle LMD

Ce modèle ci, proposé dans les articles [19] et [20] se base sur la distance locale minimum entre les objets géométriques.

La distance entre deux surfaces paramétriques $F(u,v)$ et $G(s,t)$ est égale à :

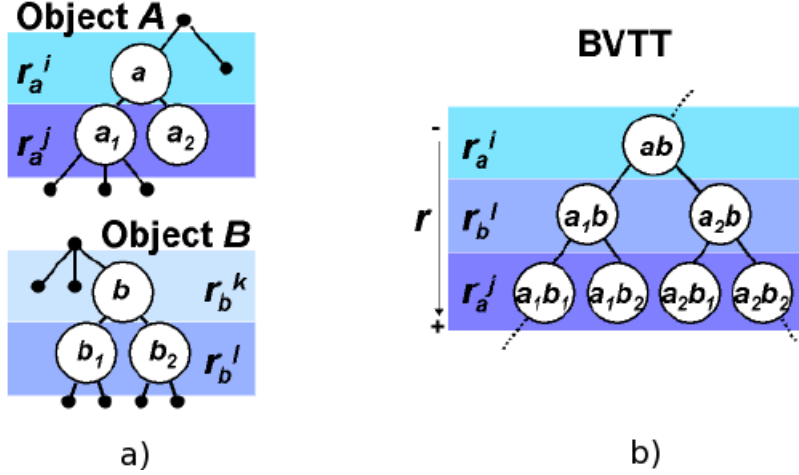


FIG. 2.5 – a) Deux hiérarchies pour deux objets A et B b) Le BVTT correspondant

$$D^2(u, v, s, t) = ||F(u, v) - G(s, t)||$$

de cette définition on peut déduire les extremums globaux en résolvant les équations suivante :

$$\begin{aligned} (F(u, v) - G(s, t)).F_u &= 0 \\ (F(u, v) - G(s, t)).F_v &= 0 \\ (F(u, v) - G(s, t)).G_s &= 0 \\ (F(u, v) - G(s, t)).G_t &= 0 \end{aligned}$$

on peut démontrer par la suite que la distance minimum globale est égale à la solution minimum de ces quatre équations.

On pourrait très bien alors se servir de cette distance minimum globale pour calculer les forces et couples de retour, cependant le fait d'avoir un seul paramètre, la distance globale, pour calculer les forces et couples de retour pourrait créer de l'instabilité, des indéterminations.

Pour résoudre ce problème, Jonhson et Willemsen, proposent d'utiliser les distances minimums locales, pour ce faire, il faut décomposer les objets géométriques afin de calculer la distance minimum locale pour chacune des parties issues de la décomposition, pour ce faire la "spatialized normal cone hierarchies data" est utilisé, cette structure est décrite dans l'article [21].

Pour ce qui est du calcul des forces et couples de réponses, à chaque cycle haptique sont calculés les LMDs entre l'objet contrôlé par l'utilisateur et le reste des objets de l'environnement qui sont inférieures à la distance minimum critique, on peut alors considérer chaque LMD comme étant un ressort virtuel avec une longueur égale à la distance critique et de raideur k, la force exercée alors entre deux points de chaque LMD_i est égal à :

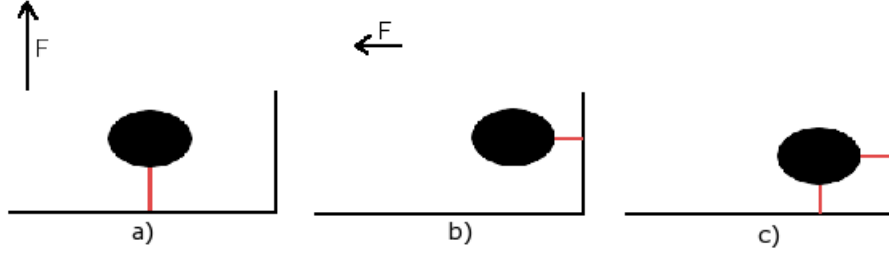


FIG. 2.6 – au passage de a) à b) la force de retour F change brusquement de sens, dans c) on est incapable de définir la distance minimum globale

$$\vec{f}_i = (distCritique - \|LMD_i\|).LMD_i$$

Et la force \vec{F} ainsi que le couple $\vec{\tau}$ de retour sont donc égal à :

$$\begin{aligned}\vec{F} &= \sum_i -k \vec{f}_i \\ \vec{\tau} &= \sum_i \vec{p}_i \times \vec{f}_i\end{aligned}$$

ou \vec{p}_i est la distance entre le centre de gravité et le point de la surface associée à la LMD_i

2.2.3 le modèle VoxMap-PointShell

Ce modèle est décrit dans l'article [22], l'idée principale de cet article est que le calcul classique d'interpénétration d'objets géométriques est beaucoup trop lent et par conséquent peu adapté, McNeely et son équipe propose alors de discrétiser ces objets afin d'accélérer ces calculs.

Définitions et principe

Leur approche est la suivante, ils décomposent régulièrement l'espace en voxels (les voxels sont des cubes de longueur n , n est suffisamment petit par rapport à la taille de l'espace), et les objets présent dans cet espace ne sont plus représentés par leurs surfaces ou leurs volumes mais par l'ensemble des voxels qui intersectent les objets (cf figure 2.7). L'objet contrôlé par l'utilisateur quant à lui est représenté par un ensemble de points disposés sur sa surface, la disposition de ces points est évidemment un point crucial de modèle, mais elle n'est pas explicité dans cet article mais un algorithme de positionnement exact des PointShells est décrit dans l'article suivant [23].

Il consiste dans un premier temps à voxeliser l'objet contrôlé par l'utilisateur puis de définir comme position initiale pour les PointShells le centre des voxels interceptant la surface de l'objet. Et dans un second temps la position exactes des PointShells est trouvé en calculant leurs projections sur la surface de l'objet.

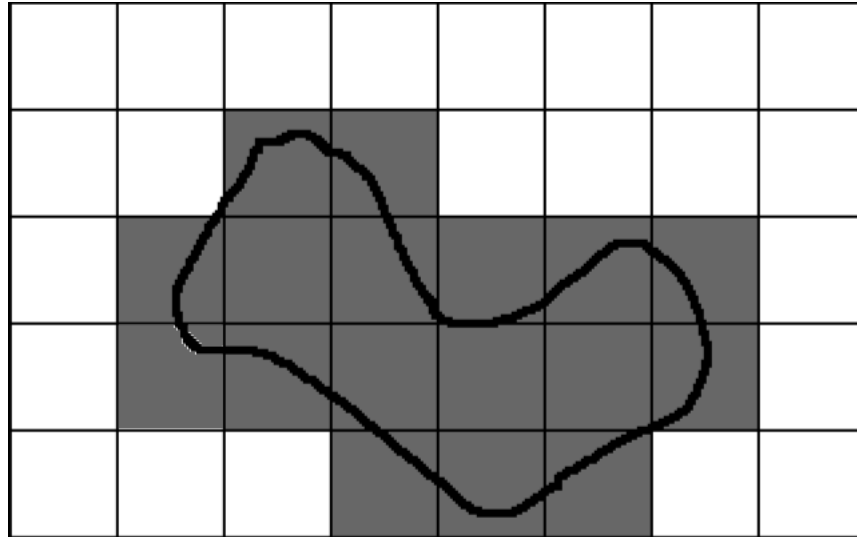


FIG. 2.7 – Un exemple de voxelisation d'un objet géométrique dans un espace à 2 dimensions(en gris les voxels des l'objet)

Cette discrétisation se fait au tout début, lors de la création de l'environnement et n'est plus refaite par la suite, la détection de collision entre l'objet contrôlé par l'utilisateur et les autres objets de l'environnement se résume donc juste à des tests d'appartenance de points à des cubes, ce qui est considérablement plus coûteux en terme de temps de calcul.

Génération des forces de retour

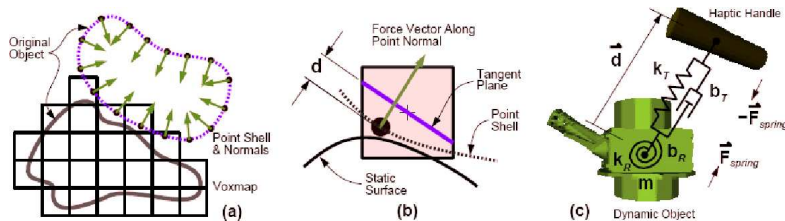


FIG. 2.8 – a) Détection d'une collision b) Zoom sur un PointShell c) couplage virtuel permettant la génération des forces de retour

Les forces et couples de collision sont calculées en fonction d'un paramètre \vec{d} . Ce paramètre est égal à la somme, pour chaque PointShell en collision avec un voxel, du vecteur entre le plan tangent à la surface de l'objet et le plan

parallèle passant par le centre du voxel responsable de la collision (cf figure 2.8 (b)). Le modèle étant basé sur le couplage virtuel (cf figure 2.8 (c)) la norme des forces et couples qui seront retournés à l'utilisateur sont donc égale à :

$$\begin{aligned}\vec{F}_{spring} &= k_T \vec{d} - b_T \vec{v} \\ \vec{\tau}_{spring} &= k_R \vec{\theta} - b_R \vec{\omega}\end{aligned}$$

ou :

k_T, b_T = la rigidité et la viscosité du ressort de translation.

k_R, b_R = la rigidité et la viscosité du ressort de rotation.

$\vec{\theta}$ = l'angle du mouvement de rotation.

$\vec{v}, \vec{\omega}$ = la vitesse de translation et de rotation de l'objet contrôlé par l'utilisateur.

Chapitre 3

UN MODELE HAPTIQUE BASE SUR LA DETECTION DE COLLISION SEGMENT-POLYGONE

3.1 Introduction

Lors de ce stage, il m'a été demandé par Eurocontrol de concevoir et d'implémenter un modèle de moteur haptique qui soit, si ce n'est plus performant en tout point de vue que les modèles existant, au moins plus adaptés au projet de remplacement des écrans radars actuels par un système de représentation en trois dimensions du trafic aérien augmenté d'un moyen de contrôle haptique. Il fallait donc que ce modèle ait six degrés de liberté, et possède un temps de réaction très court, la génération des forces peut également être basé sur la profondeur de la pénétration, celle la n'étant pas interdite tant qu'elle est raisonnable bien sur.

3.2 Principe

J'ai donc choisi un modèle de représentation discrète des objets contrôlés par les utilisateurs, un peu comme dans le modèle VPS de McNeely, l'objet contrôlé par l'utilisateur est représenté par une ensemble de points significatifs (appelé SignificantPoint), la différence avec le VPS se situe sur le fait que ces points ne sont pas nécessairement placés sur la surface de l'objet. En effet, depuis que l'interpénétration des objets est autorisée, il se peut que certaines zones soient plus importantes que d'autres et par conséquent qu'il faille plus les protéger de l'inter pénétration (Ce point sera étudié plus en détails dans une des sections

suivantes.)



FIG. 3.1 – Un avion modélisé en 3D et ses SignificantPoints

Les autres objets de l'environnement quand à eux sont représentés par des polyèdres, on appelle polyèdre l'union fermé de polygones convexes.

Entre deux cycles du moteur haptique, si il y a mouvement de l'objet contrôlé par l'utilisateur, on peut définir des segments entre l'ancienne position des SignificantPoints et la nouvelle, et au cas ou il y aurait une collision entre l'objet contrôlé par l'utilisateur et un des objets de l'environnement celle-ci pourrait être détectée en cherchant si il y avait une intersection entre un ou (plusieurs) des segments et un (ou plusieurs) des polygones de l'objet de l'environnement.

3.2.1 La détection de collision

Pour détecter ces collisions, pour chacun des SP, nous définissons tous d'abord un FLAG représentant l'état de ce SP, ce FLAG peut prendre pour valeur : *collided*, *notcollided* et nous définissons aussi un point de collision (CP). Ceci fait si nous considérons qu'à l'état initial l'objet contrôlé par l'utilisateur n'est en collision avec aucun des objets de l'environnement, nous mettons le FLAG de chaque SP à la valeur *notcollided* et le CP à NULL.

Par la suite imaginons que après un mouvement il y ait une collision, alors nous obtenons pour au moins un SP le cas décrit par la figure 3.2, SP_0 , SP_1 , SP_2 , SP_3 et SP_4 étant les positions successives que prend le SP après chaque mouvement, nous définissons alors le CP de ce SP comme étant le point d'intersection du segment formé par celui-ci lors du déplacement de l'objet et le vecteur \vec{d}_n comme le vecteur allant de CP à SP_n .

Il semble évident que tant que le produit scalaire entre la normale \vec{n} du polygone et le vecteur \vec{d}_n est négatif, l'objet contrôlé par l'utilisateur est en collision avec l'objet de l'environnement auquel appartient ce polygone, il est donc inutile pour ce SP tant que ce produit scalaire est négatif de détecter s'il est en collision avec un autre polygone. En d'autre terme, sur la figure 3.2 il

n'est pas nécessaire de tester si il y a une intersection entre les polygones et les segments formés par les déplacements jusqu'aux positions SP_2 et SP_3 , il suffit donc lorsqu'une collision est détectée sur un SP de mettre la valeur de son FLAG à *collided*, de mettre celui-ci à la valeur *notcollided* lorsque le SP sort de l'objet, autrement dit, quand le produit scalaire entre \vec{d}_n et \vec{n} est négatif, puis de ne tester si il y a une collision impliquant un SP qui si son FLAG a pour valeur *notcollided*.

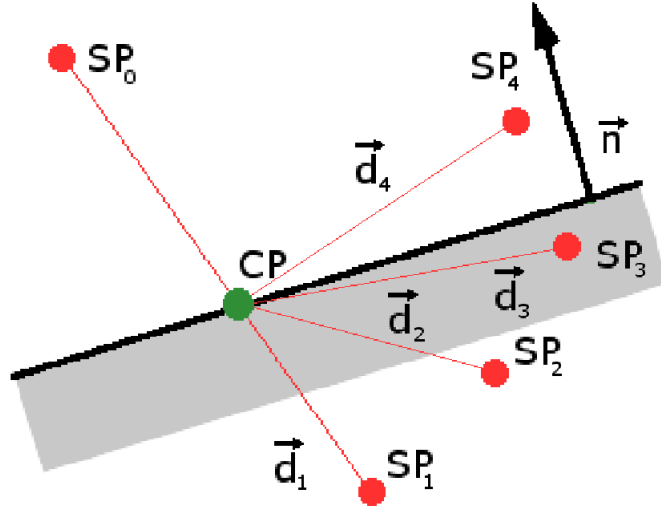


FIG. 3.2 – Collision d'un SP avec une objet de l'environnement, 5 étapes.

Ce qui nous donne si la structure d'un SP est la suivante :

L'algorithme de détection de collision (pour un SP) en pseudo code suivant :

3.2.2 Génération des forces et des couples de réponse

Une fois la collision détectée, il reste encore à calculer les forces de retour et couples de retour.

Supposons G (cf. figure 3.3) la position du centre de gravité de l'objet contrôlé par l'utilisateur, soit pour chaque SignificantPoint (SP) \vec{GSP} la position relative du SignificantPoint par rapport à la position du centre de gravité de l'objet contrôlé par l'utilisateur.

La force de retour pour SP, \vec{F}_{SP} , est donc égale à :

$$\vec{F}_{SP} = -(\vec{d} \cdot \vec{n}) \times \vec{n} \times K_f$$

SignificantPoint structure :

HOPtr : Pointeur	Pointeur sur l'objet auquel appartient le SP
CurrentPosition : Vector # initialisé à (0,0,0)	Vecteur représentant la position actuelle
LastPosition : Vector # initialisé à (0,0,0)	Vecteur représentant la dernière position
FLAG : boolean # initialisé à <i>notcollided</i>	Booléen représentant l'état du SP
CollisionPoint : Vector # initialisé à NULL	Vecteur représentant la point de collision
FBDirection : Vector # initialisé à NULL	Normale du polygone en intersection
FBF : Vector # initialisé à (0,0,0)	Force de retour
CBF : Vector # initialisé à (0,0,0)	Couple de retour

Et le couple de retour pour SP, $\overrightarrow{C_{SP}}$, est donc égale à :

$$\overrightarrow{C_{SP}} = -(\vec{d} \times \vec{n}) \times K_f$$

ou :

\vec{d} est le vecteur entre le point de collision et la position actuelle du SP

\vec{n} est la normale du polygone en collision avec le SP

K_f une constante caractérisant la rigidité de l'objet en collision avec le SP.

Nous expliquerons et nous justifierons le choix de ce calculs de forces et de couples dans la partie concernant implémentation de ce modèle.

3.2.3 Analyse des avantages et des défauts de ce modèle

Le principal défaut de cette représentation est le suivant, à cause de la discrétisation des objets contrôlés par l'utilisateur, certaines collisions ne pourront pas être détectée, par contre cette approche règle deux problèmes inhérent aux modèles dont le calcul de force est basé par la profondeur de la pénétration, la discontinuité des forces et le fait de pouvoir passer à travers des objets fins.

En effet, une des conséquence de la discrétisation est que suivant le degrés de cette discrétisation (c'est à dire l'espace entre deux SPs consécutifs), il peut arriver qu'un polygone soit suffisamment fin pour pouvoir être en collision avec

SPCollision

Input :

SP : SignificantPoint SignificantPoint
NewPosition : Vector Nouvelle position pour le SP
Environnement Polygon[1..n] Environnement

Begin :

```
SP.LastPosition ← SP.CurrentPosition ;
SP.CurrentPosition ← NewPosition ;

if (SP.FLAG = notcollided)
  for( $i = 1 ; i < n ; i \leftarrow i + 1$ ) faire
    S ← Segment (SP.LastPosition,SP.CurrentPosition) ;
    if (S ∩ Environnement[i])
      SP.FLAG = collided ;
      SP.CollisionPoint ← S ∩ Environnement[i]
      SP.FBDirection ← S Environnement[i].normal
    endif ;
  endfor ;
endif ;

if (SP.FLAG = collided)
  SP.FBF ← GenerateForce(SP) ;
  SP.CBF ← GenerateTorque(SP) ;
  d ← SP.CurrentPosition - SP.LastPosition ;
  if (d . (SP.FBDirection) > 0)
    SP.FLAG = notcollided ;
  endif ;
endif ;
```

l'objet sans pour autant être en collision avec un SP (cf. 3.4 a), ce problème est aussi un problème que l'on retrouve dans le modèle de McNeely.

Par contre un des avantages de ce modèle par rapport à d'autres est qu'il règle le problème de discontinuité de force au niveau des SP. Dans les modèles où les objets sont représentés par un ensemble de points (ou bien même les modèles de rendu haptique pour 3DOF), la plupart recherche, quand un SP (ou leurs équivalents dans les autres modèles) est en collision avec un objet de l'environnement, quel est le point de la surface de l'objet le plus proche du SP et tendent

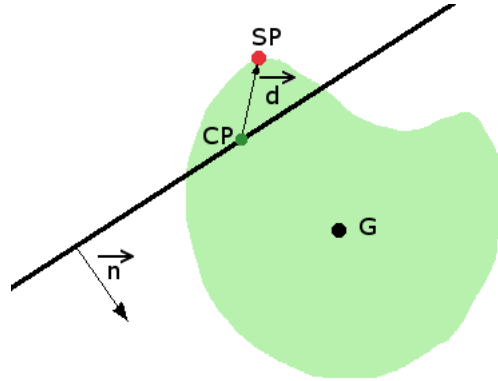


FIG. 3.3 – Objet en collision avec un polygone de normale n

à ramener le SP vers celui-ci.

Or si on regarde la figure 3.4 b il se peut que ce point suivant le mouvement du SP change brusquement, on obtient alors concernant ce SP un brusque changement de force. Le modèle que je propose règle ce problème là en ramenant toujours le SP à la première collision détectée, ou tout du moins au point le plus proche de la surface où s'est produite la collision. Et c'est avec le même système de calcul de force que l'on évite de passer à travers des objets trop fins lors d'un trop grand mouvement de la part de l'utilisateur, les modèles qui détectent les collisions en cherchant si le SP est à l'intérieur d'un objet de l'environnement peuvent manquer une collision si entre deux position du SP il y a des objets suffisamment minces (cf. 3.4 c).

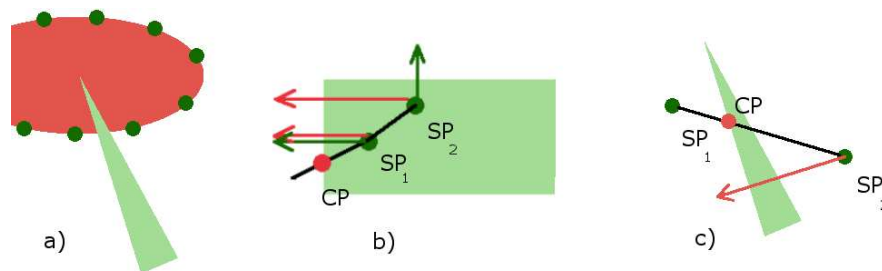


FIG. 3.4 – a) Problème dû à la discrétisation b) et c) résolution de deux problèmes récurrents en haptique.

3.3 Implémentation

L'implémentation de modèle est semble-t-il au premier abord d'une extrême simplicité, mais hélas, lors de la première implémentation de celui-ci, je me suis retrouvé confronté à un problème important, comment définir la constante de rigidité K_f pour faire en sorte que lors d'une collision avec un objet de l'environnement l'objet contrôlé par l'utilisateur sorte de l'objet avec lequel il est en collision.

La première idée fut de mettre une constante extrêmement forte, les forces de retour rendues à l'utilisateur devraient faire en sorte que l'objet contrôlé par celui-ci ne soit plus aucunement en collision. Cette approche fonctionne très bien pour ce qui est de revenir à un état sans collision le plus vite possible, par contre elle rend impossible l'utilisation de ce système tant il devient instable et tant les forces et couples sont bien trop fortes.

L'idéal aurait été que les forces et couples de retour soient d'intensité égale et de direction opposée à celles de la composante des forces appliquées par l'utilisateur sur le bras selon la normale du polygone en collision, le problème est que les bras haptique de nos jours ne disposent pas de capteurs d'efforts, il est donc impossible de connaître celles-ci.

Ne pouvant déduire par le calcul la valeur de K_f pour laquelle le système est le plus agréable à utiliser, car n'ayant trouvé aucun modèle mathématique permettant de mesurer le degré d'aisance et de facilité d'utilisation d'un tel système, j'ai du mener une études sur des sujets humains.

L'expérience consistait à faire tester à plusieurs utilisateur mon modèle haptique, dans un environnement ou les seules contraintes étaient un cube, les utilisateurs quant à eux contrôlaient une petite sphère composé de 64 SPs, les utilisateurs ont également à leur dispositions un moyen de faire varier la valeur de K_f (sans pour autant voir sa valeur). Une fois que l'utilisateur à trouvé la valeur de K_f pour laquelle le système lui semble le plus agréable et exact à utiliser, il appuie sur une touche du clavier prédéterminée, après pression de cette touche, l'ordinateur enregistre la valeur de K_f pour cet utilisateur ainsi que la moyenne du nombre de SPs en collision lorsqu'il y a eu collision.

Les résultats obtenues sont les suivant :

Sujets	1	2	3	4	moy.	var.	$\frac{var.}{moy.}$ (%)
K_f	0.062	0.084	0.07	0.112	0.082	0.016	19.5
nb. moy. SP	8.01	8.56	9.01	6.21	7.95	0.62	7.98
$K_f \times$ nb. moy. SP	0.497	0.719	0.631	0.696	0.636	0.078	12.2

Bien que cette étude par manque de temps se soit faite à partir d'un très petit échantillons d'individus, nous pouvons néanmoins remarquer qu'il semblerait que la valeur de K_f ne soient pas d'une première importance (d'après le pourcentage variance par moyenne) mais que par contre la valeur $K_f \times$ nb. moy. SP est déjà plus commune à tout les membres de l'échantillon testé.

Une autre critique qui pourrait être faite à cette expérimentation est la suivante, le choix d'une sphère comme objet contrôlé par l'utilisateur n'est pas forcément

judicieux, en effet avec une sphère, il est impossible de se rendre compte de l'importance de la valeur de K_f pour les couples, vu qu'ils sont inexistant (en fait du à la discrétisation de la sphère, il y'en a quelques uns quand même, mais leur intérêt est nul vu que ce ne sont que des conséquences des défauts du modèle et qu'ils ne sont de plus quasiment pas perceptibles).

Il semble évident qu'une étude plus approfondie pourrait et devrait être faite, mais elle ne le sera pas effectué lors de ce stage, par manque de temps et aussi par que cette étude n'apporterait rien de plus pour ce qui concerne la validité du modèle.

Pour toute la suite de l'implémentation est des tests, nous prendrons 0.082 pour valeur de K_f .

Une fois le problème de la valeur de K_f résolu, se pose immédiatement un autre problème, avec cette valeur les forces et couples ne sont pas suffisant pour en cas de collision faire ressortir complètement l'objet contrôlé par l'utilisateur de l'objet en collision, il reste une très légère interpénétration des deux objets. Ceci n'est aucunement détectable par l'utilisateur mais pose un problème pour détecter la fin de la collision vu que le produit scalaire entre \vec{d}_n et \vec{n} sera toujours négatif (cf. partie sur le principe), et tout polygone en collision avec un SP sera perçu comme un plan et non comme un polygone (voir fig a). Ce problème sera résolu lorsque la technologie des bras haptique permettra de connaître la force appliquée par l'utilisateur.

En attendant, une des autres solutions possible et de rajouter une condition pour considérer la sortie de collision d'un SP, un SP qui était en collision ne l'est plus si il traverse une autre parois de l'objet (voir fig 3.6 b).

Il nous faut donc remplacer la partie de l'algorithme SPCollision traitant de la sortie de collision du SP par le pseudo code :

```
SPCollision

...

if (SP.FLAG = collided)
  SP.FBF ← GenerateForce(SP);
  SP.CBF ← GenerateTorque(SP);
  d ← SP.CurrentPosition - SP.LastPosition;
  S ← Segment (SP.CurrentPosition,SP.LastPosition);
  if (d . (SP.FBDirection) > 0) or ((S ∩ CollidedObject[i])
    SP.FLAG = notcollided;
  endif;
endif;
```

Cette solution peut paraître très lourdes, car elle rajoute des tests d'intersections entre un segment mais cela reste somme toute raisonnable car il ne s'agit

que de tester sur quelques polygones, ceux de l'objet en collision et non pas tous ceux de l'environnement.

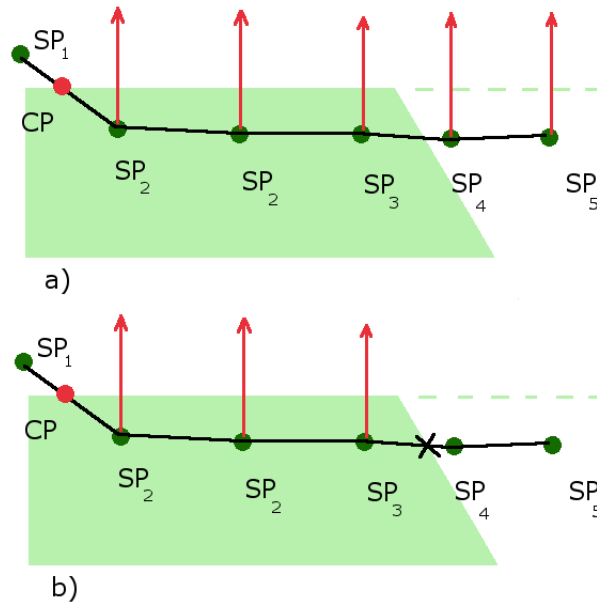


FIG. 3.5 – a) Objet continu, problème du à la génération de force par profondeur de pénétration b) Solution au problème

3.4 Un nouvel algorithme pour la détection d'intersection segment-polygone

3.4.1 Introduction

La détection de collision dans mon moteur haptique étant basée sur l'intersection ou non de segments avec des polygones, il était donc important d'avoir un algorithme pour calculer ces intersections. Les deux algorithmes habituellement utilisés pour déterminer si un segment intercepte un polygone sont ceux de Badouel [24] et de Möller-Trumbore [25], nous commencerons donc par en étudier les principes. Puis je proposerai par la suite un nouvel algorithme basé sur le sens de rotation.

Dans tous le reste de cette partie, un segment $R(t)$ sera défini par son origine O et son vecteur directeur \vec{D} tel que :

$$R(t) = O + t\vec{D}$$

et un polygone est composé de l'ensemble de ses arrêtes V_0, \dots, V_n .

3.4.2 L'algorithme de Badouel

L'algorithme de Badouel pour détecter l'intersection d'un segment et d'un polygone commence par décomposer le polygone en triangle. Par la suite il détermine si le segment intercepte le plan du polygone, et le cas échéant détermine le point de collision du segment et du plan du polygone. Ensuite l'algorithme cherche à savoir si le point de collision est à l'intérieur de l'un des triangles du polygone. Pour ce faire, il calcule pour chaque triangles du polygone les coordonnées barycentriques du point de collision. Soit $V_0V_1V_2$ un triangle du polygone et P le point de collision, on a alors l'équation 1.

$$\overrightarrow{V_0P} = \alpha \overrightarrow{V_0V_1} + \beta \overrightarrow{V_0V_2} \quad (1)$$

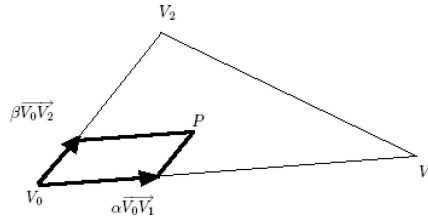


FIG. 3.6 – Coordonnées barycentriques d'un point dans un triangle

Le point P sera dans le triangle si :

$$\alpha \geq 0, \beta \geq 0, \alpha + \beta \leq 1.$$

On calcule α et β de la manière suivante :

On appelle $\vec{V}(x)$ (*resp.* $\vec{V}(y)$, $\vec{V}(z)$) la coordonnées de \vec{V} selon l'axe des x (*resp.* y, z).

$$\alpha = \frac{\det \begin{pmatrix} \overrightarrow{V_0P}(y) & \overrightarrow{V_0V_2}(y) \\ \overrightarrow{V_0P}(z) & \overrightarrow{V_0V_2}(z) \end{pmatrix}}{\det \begin{pmatrix} \overrightarrow{V_0V_1}(y) & \overrightarrow{V_0V_2}(y) \\ \overrightarrow{V_0V_1}(z) & \overrightarrow{V_0V_2}(z) \end{pmatrix}} \quad (2)$$

$$\beta = \frac{\det \begin{pmatrix} \overrightarrow{V_0V_1}(y) & \overrightarrow{V_0P}(y) \\ \overrightarrow{V_0V_1}(z) & \overrightarrow{V_0P}(z) \end{pmatrix}}{\det \begin{pmatrix} \overrightarrow{V_0V_1}(y) & \overrightarrow{V_0V_2}(y) \\ \overrightarrow{V_0V_1}(z) & \overrightarrow{V_0V_2}(z) \end{pmatrix}} \quad (3)$$

3.4.3 L'algorithme de Möller-Trumbore

L'algorithme de Möller-Trumbore est, d'après ses auteurs, plus rapide que l'algorithme étudié précédemment (sans précalculs), son principe est pourtant similaire, décomposition du polygone en triangles puis calculs des coordonnées barycentriques. La différence se fait au niveau du calcul de ces coordonnées, pour ce faire Möller et Trumbore font subir au triangle une série de transformations géométriques dans le but d'aligner deux des cotés du triangle avec les axes des coordonnées.

Soit $M = [-\vec{D}, \vec{V}_0\vec{V}_1, \vec{V}_0\vec{V}_2]$, α, β les coordonnées barycentriques, O le point d'origine du segment et t le paramètre de son équation, après les transformations les calculs s'en trouvent grandement simplifiés, il en résulte les équation (4) et (5). Le segment R intercepte le triangle $V_0V_1V_2$ si :

$$M \begin{bmatrix} t \\ \alpha \\ \beta \end{bmatrix} = \vec{V}_0\vec{O} \quad (4)$$

et donc si :

$$\begin{bmatrix} t \\ \alpha \\ \beta \end{bmatrix} = \frac{1}{(-\vec{D} \wedge \vec{V}_0\vec{V}_2) \cdot \vec{V}_0\vec{V}_1} \begin{bmatrix} (\vec{V}_0\vec{O} \wedge \vec{V}_0\vec{V}_1) \cdot \vec{V}_0\vec{V}_2 \\ (\vec{D} \wedge \vec{V}_0\vec{V}_2) \cdot \vec{V}_0\vec{O} \\ (\vec{V}_0\vec{O} \wedge \vec{V}_0\vec{V}_1) \cdot \vec{D} \end{bmatrix} \quad (5)$$

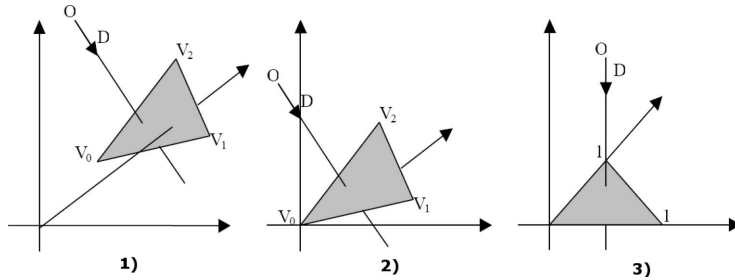


FIG. 3.7 - Transformations géométriques dans l'algorithme de Möller-Trumbore : 1) Position initiale 2) Translation jusqu'à l'origine, 3) Rotation et mise à l'échelle par $M^{-1}[V_0]$

3.4.4 Algorithme du sens de rotation

Cet algorithme utilise des concepts complètement différents, en tout premier lieu il est important de rappeler qu'il ne marche que sur les polygones convexes. Son principe est le suivant, soit un polygone dont les sommets sont $V_0V_1V_3\dots V_n$,

alors un point P est dans le polygone si et seulement si :

Soit ρ la fonction qui a une rotation associe son sens alors, $\forall i \in \{1, \dots, n\}$ on a

$$\rho \left(\langle \overrightarrow{PV_i}, \overrightarrow{PV_{i+1 \pmod n}} \rangle \right) = \rho \left(\langle \overrightarrow{V_j V_i}, \overrightarrow{V_j V_{i+1 \pmod n}} \rangle \right)$$

avec $j \in \{1..n\} \setminus \{i, i+1 \pmod n\}$.

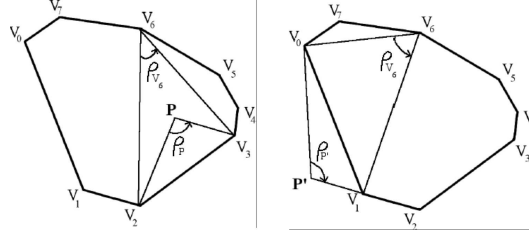


FIG. 3.8 – Exemple du principe de l’algorithme du sens de rotation : P est dans le polygone car $\rho_{V6} = \rho_P$, P’ n’est pas dans le polygone car $\rho_{V6} \neq \rho_{P'}$

L’algorithme se déroule alors de la façon suivante, tout d’abord il détermine si le segment intercepte le plan du polygone, et le cas échéant détermine le point de collision du segment et du plan du polygone. Ensuite il vérifie que ce point appartient au polygone de la façon suivante :

$$\begin{aligned} \vec{N} &= \overrightarrow{V_0 V_1} \wedge \overrightarrow{V_0 V_2} \\ \vec{K}_i &= \overrightarrow{PV_i} \wedge \overrightarrow{PV_{i+1 \pmod n}} \end{aligned}$$

P est dans le polygone si :

$$\vec{N} \cdot \vec{K} > 0$$

Car si le sens de rotation est le même, alors la direction du produit vectoriel est la même, il faudrait aussi prendre comme précaution que $P \notin V_0 V_1$. On peut ensuite s’éviter de faire d’autres calculs en ne comparant qu’une seule des composantes non nulles de ces vecteurs. Ce qui nous donne l’équation (6).

$$j \in \{x, y, z\} \text{ et } \vec{N}(j) \neq 0$$

$$\vec{K}(i, j) = \begin{cases} \overrightarrow{PV_i}(y) \times \overrightarrow{PV_i \pmod n}(z) - \overrightarrow{PV_i}(z) \times \overrightarrow{PV_i \pmod n}(y) & \text{si } j = x \\ \overrightarrow{PV_i}(z) \times \overrightarrow{PV_i \pmod n}(x) - \overrightarrow{PV_i}(x) \times \overrightarrow{PV_i \pmod n}(z) & \text{si } j = y \\ \overrightarrow{PV_i}(x) \times \overrightarrow{PV_i \pmod n}(y) - \overrightarrow{PV_i}(y) \times \overrightarrow{PV_i \pmod n}(x) & \text{si } j = z \end{cases}$$

Si P est un point du polygone on a :

$$\forall i \in \{1, \dots, n\} \vec{N}(j) \times \vec{K}(i, j) > 0 \quad (6)$$

3.4.5 Comparaison du coût de ces algorithmes

En tenant compte du fait que les polygones sont fixes, on peut pour les équations (2) et (3) précalculer :

$$\frac{1}{\det \begin{pmatrix} \overrightarrow{V_0V_1}(y) & \overrightarrow{V_0V_2}(y) \\ \overrightarrow{V_0V_1}(z) & \overrightarrow{V_0V_2}(z) \end{pmatrix}}$$

Il ne reste donc qu'à calculer pour l'algorithme de Badouel deux déterminants, ce qui coûte 4 multiplications et deux additions, et de les multiplier par les valeurs précalculées, soit pour chaque triangle du polygone un coût de six multiplications, trois additions et trois comparaisons.

Pour ce qui est de l'équation (5), on ne peut hélas rien précalculer car tout dépend des segments et eux ne sont pas fixes. Le coût du test de Möller-Trumbore est donc de quatre produits vectoriels, quatre produits scalaires, trois divisions, une addition et trois comparaisons. Soit en tout trente-six multiplications, vingt-et-une additions, trois divisions et trois comparaisons et ce par triangle.

On peut en revanche pour l'équation (6) précalculer $\vec{N}(j)$, il ne reste donc à calculer que $\vec{K}(i, j)$, ce qui se fait en deux multiplications et une addition. Le coût du test du sens de rotation est donc de trois multiplications, une addition et une comparaison par coté du polygone.

Si l'on tient compte du fait que un polygone convexe de n cotés peut se décomposer en au minimum $(n - 2)$ triangles et que le coût en nombre de cycles des opérations usuelles est le suivant :

Opération	ADD	MUL	DIV	CMP
Nb. cycles	1	8	40	1

On obtient alors :

$$\begin{aligned} \text{Coût de Badouel}(n) &= 54n - 13. \\ \text{Coût de Möller-Trumbore}(n) &= 432n - 864. \\ \text{Coût de Test de rotation}(n) &= 26n + 95. \end{aligned}$$

Möller-Trumbore est si on tient compte des précalculs est donc de façon évidente l'algorithme le moins performant des trois. Il reste donc à comparer le test de Badouel et celui de la rotation.

$$\begin{aligned} \text{Badouel}(n) &> \text{rotation}(n) \\ 54n - 13 &> 26n + 95 \\ 28n &> 108 \\ n &> 108/28 \\ n &> 3.8 \end{aligned}$$

Donc si les collisions à tester se font entre des segments et des triangles Badouel reste le meilleur algorithme, pour tout les autres polygones le test de rotation semble meilleur. (Il faut aussi faire attention au fait que cette étude théorique ne prend pas en compte l'exécution en parallèle de plusieurs instructions).

3.4.6 Vérification expérimentale

Afin de vérifier les résultats théoriques obtenus, j'ai réalisé une série de tests de collision, 2^{26} avec chaque algorithmes et pour chaque moyenne de nombre de cotés des polygones allant de 3 à 10. Ces tests ont été effectués sur un ordinateur équipé d'un processeur cadencé à 1,5GHz et disposant de 500Mo de mémoire vive.

Les résultats obtenus se ci dessous (voir figure 3.9), sur l'axe des abscisses se trouvent la moyenne du nombre de cotés par polygone, et sur l'axe des ordonnées le temps en seconde mis pour réaliser l'ensemble des tests de collisions.

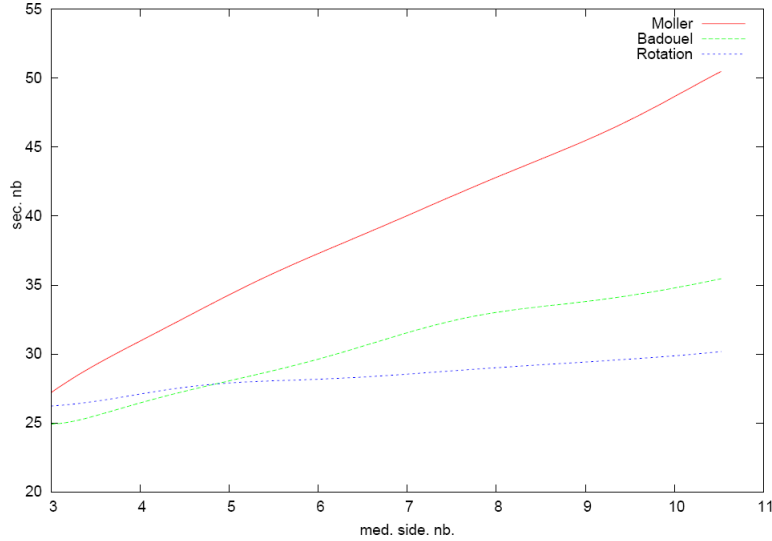


FIG. 3.9 – Résultats expérimentaux, en abscisses le nombre moyen de cotés pour les polygones des tests de collision et en ordonnées le nombre de seconde mis pour réaliser ces tests

Les résultats obtenus ne sont pas véritablement ceux qui étaient attendus là ou les résultats théoriques annonçait que mon algorithme était plus performant que celui de Badouel si les polygones avaient plus de trois cotés (3.8 en fait), les résultats pratiques eux montrent que l'algorithme de Badouel reste plus performant pour des polygones dont le nombre de cotés peut atteindre 4 (4.7).

Cette différence entre la théorie et la pratique peut peut-être s'expliquer de par le fait que dans le cas de Badouel, lorsqu'une collision est détectée, elle n'est pas forcément détectée sur le dernier triangle du polygone, de même pour ce qui est de mon algorithme, en cas de non détection, il suffit seulement que un des produits scalaires soient négatif, si cela se produit sur le premier alors le test sera extrêmement rapide, ceci peut expliquer la différence et dans ce cas une étude théorique de la complexité moyenne ou probabiliste et non plus de la complexité dans le pire des cas donnerait de meilleurs résultats, on peut

peut-être aussi associer cette erreur au fonctionnement interne des processeurs (parallélisme, optimisations...)

3.5 Performances

Pour tester les performances de ce modèle haptique, j'ai procédé à une expérimentation. Cette expérimentation était la suivante, elle consistait à générer successivement des environnements composés d'un nombre croissant de triangles dans lesquels évoluera un objet dynamique composé de 100 SP et de relever pour chacun d'eux le taux de rafraîchissement.

Les mouvements de l'objet haptique ont été générés automatiquement et ne sont pas engendrés par un utilisateur, pour éviter les erreurs pouvant être du à un quelconque facteur humain.

Ces tests ont été effectués sur une machine équipée d'un processeur AMD cadencé à 1,5GHz et disposant de 500Mb de RAM, le nombre de frames ayant servi à calculer les FPS pour chaque environnement est de 10^6 (cela correspond à un temps d'exécution allant de 17 à minutes 42 minutes par environnement).

Les résultats obtenues se trouvent exposés dans la courbe ci-dessous (voir figure 3.10), sur l'axe des abscisses se trouvent le nombre de polygone composant l'environnement et sur celui des ordonnées le nombre de Frame Par Seconde (FPS) du rendu haptique pour cet environnement.

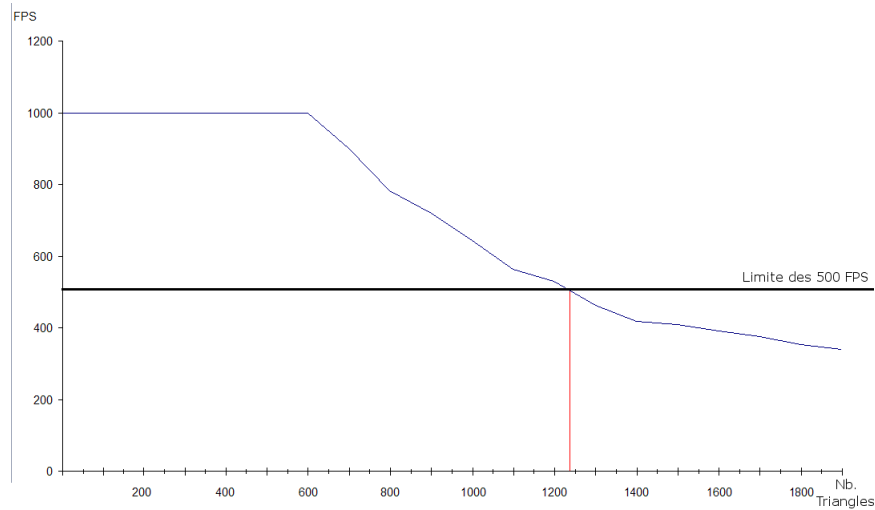


FIG. 3.10 – Nombre de FPS en fonction du nombre de polygones de l'environnement

On peut déduire de cette courbe que le taux de rafraîchissement idéal, 1000 FPS, est atteint pour des environnement comprenant jusqu'à 600 triangles. On

peut expliquer la stagnation (partie linéaire jusqu'à 600 triangles par environnement) par la limitation du bras haptique, la technologie des servo-moteurs des bras PHANToM ont un taux de rafraîchissement limite de 1000 FPS.

Pour les environnement possédant plus de 600 triangles, les performances ne sont plus optimales et décroissent de façon logarithmique. Si l'on considère que le taux de rafraîchissement est acceptable pour une zone allant de 1000 à 500 FPS, le rendu de mon moteur haptique est efficace pour des environnements composé de 1230 triangles.

Il faut donc pour les environnement composé de plus de 1200 triangles trouver une autre solution. Une solution efficace pourrait être le partitionnement de l'espace. Le partitionnement de l'espace consiste comme son nom l'indique à fragmenter l'environnement pour n'avoir par la suite qu'à traiter les détections de collisions uniquement que dans la fraction de l'environnement où se trouve l'objet contrôlé par l'utilisateur.

Parmi les algorithmes de partitionnement possible, les plus habituellement utilisés sont les octrees [26], les BSP-Tree [27] et les algorithmes basé sur des tables de hachages. Chacune de ces méthodes consomment plus ou moins de mémoires et sont plus ou moins efficaces en fonction des environnements, mais leurs études sort du cadre de ce rapport. Il est juste important de retenir que ces méthodes permettraient à moindre coût de réduire le nombre de tests de collision et par conséquent amélioreraient les performances en permettant de définir des fraction d'environnement ne contenant au maximum 600 triangles.

Chapitre 4

PETIT DETOUR PAR L'ALGEBRE GEOMETRIQUE

4.1 Préambule

Lors de mes lectures sur les différents algorithmes haptiques et géométriques, j'ai été assez surpris de m'apercevoir qu'il n'y avait pas de théorie générale pour décrire la géométrie euclidienne. Les algorithmes de détection actuels sont la plupart du temps du bricolage et ne s'appliquent que pour un unique type d'objet la plupart du temps, par exemple intersection entre un triangle et un segment, mais je n'ai pas vu d'algorithme d'intersection polygone-polygone .

J'ai donc recherché une théorie plus générale qui donnerait une description globale de la géométrie, je suis alors tombé sur l'algèbre géométrique. Même si je n'ai pu tirer une application pour le problème qui m'était donné lors de ce stage (car cette algèbre permet de traiter uniquement les objets infini, droite, plan ...), j'ai trouvé tout de même intéressant d'en parler, tout d'abord car cela m'a pris un certain temps pour l'assimiler et qu'elle pourrait me resservir et deuxièmement car elle est très élégante.

4.2 Histoire de l'algèbre géométrique

En 1882 dans un papier intitulé "On the classification of geometric algebras" [28] William K. Clifford introduit pour la première fois le concept d'algèbre géométrique. Il réalisa que le produit extérieur de Grassmann et que les quaternions pouvaient être réunis dans une même algèbre au prix d'un léger changement dans la définition du produit extérieur, il introduisit donc le produit géométrique, que nous étudierons plus tard.

Depuis l'algèbre géométrique a trouvé une utilité dans de nombreux domaines scientifiques, David Hestenes dans les années 80 fut le premier à en montrer les avantages pour la physique, la mécanique et la géométrie [29][30][31]. D'autres applications lui furent trouvées par la suite notamment en mécanique et en physique quantique.

La première apparition de l'algèbre géométrique en informatique se fit aux conférences de SIGGRAPH 2000 et SIGGRAPH 2001, elle fut par la suite utilisée dans des films d'animation, notamment par Pixar.

4.3 Introduction mathématique à la géométrie algébrique

4.3.1 Espaces vectoriels

Un espace vectoriel est un ensemble de vecteurs muni d'une loi d'addition interne, celle-ci est commutative est associative et possède un élément neutre.

Soit V un espace vectoriel, a et b deux éléments de V on a alors :

$$\begin{aligned} a + b &= b + a \text{ (symétrie)} \\ a + (b + c) &= (a + b) + c \text{ (associativité)} \\ \exists e \in V \mid e + a &= a \text{ (élément neutre)} \end{aligned}$$

Un espace vectoriel est aussi muni d'une loi de multiplication externe par un scalaire qui obéit aux règles suivantes (a et b vecteurs et λ, μ scalaires) :

$$\begin{aligned} \lambda(a + b) &= \lambda a + \lambda b \text{ (distributivité à gauche)} \\ (\lambda + \mu)a &= \lambda a + \mu a \text{ (distributivité à droite)} \\ (\lambda\mu)a &= \lambda(\mu a) \text{ (associativité)} \end{aligned}$$

Si $1 \cdot a = a \forall a$ alors $1a = a, \forall a \in V$ (élément neutre)

A tout espace vectoriel fini est associé une dimension, on appelle base génératrice d'un espace vectoriel de dimension d tout ensemble de d vecteurs linéairement indépendants, il n'y a pas unicité de la base. Tout vecteur d'un espace vectoriel fini, peut s'écrire pour chaque base de l'espace comme étant une combinaison linéaire des vecteurs composant celle-ci.

Ce rappel sur les espaces vectoriel ne constitue vraiment qu'une introduction sommaire, mais elle sera suffisante pour comprendre le reste de cette partie.

4.3.2 Le produit intérieur (Inner product)

Le produit intérieur est ce que l'on appelle couramment le produit scalaire, soit V un espace vectoriel, on dit qu'une application φ :

$$\begin{aligned} V \times V &\rightarrow \mathbb{R} \\ (a, b) &\mapsto r \end{aligned}$$

est un produit scalaire si elle obéit aux règles suivantes :

$$\begin{aligned} a \varphi b &= b \varphi a \text{ (symétrie)} \\ a \varphi a &\geq 0 \text{ (positivité)} \\ a \varphi a &= 0 \Rightarrow a = 0 \end{aligned}$$

4.3.3 Le produit extérieur (Outer product)

Le produit extérieur est une généralisation du produit vectoriel pour des espaces vectoriels de dimensions supérieures à 3. Il possède les propriétés suivantes :

$$\begin{aligned} a \wedge a &= 0 \\ a \wedge b &= -b \wedge a \text{ (antisymétrique)} \\ a \wedge (b + c) &= a \wedge b + a \wedge c \text{ (distributivité)} \\ (\lambda a) \wedge b &= \lambda(a \wedge b) \end{aligned}$$

On appelle bivecteurs (trivecteurs, ...) la résultante du produit extérieur de deux (trois, ...) vecteurs linéairement indépendants, et on peut associer à chaque bivecteur (trivecteur, ...) une aire (volume, ...) orientée (voir figure 4.1).

Il faut bien faire attention, même si cela ne semble pas évident sur la figure un bivecteur ne définit en aucune manière une forme.

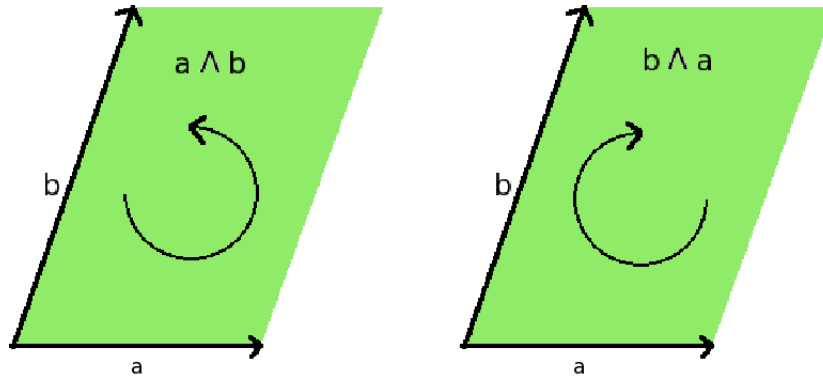


FIG. 4.1 – Chercher un titre

Pour donner une idée plus précise de la façon dont on calcul un produit extérieur, nous allons faire un exemple à partir d'un espace vectoriel à de dimension 2.

Soit V un espace vectoriel de dimension 2, alors il existe une base (e_1, e_2) de V , tel que tout vecteur de V soit une combinaison linéaire de cette base. Soit $a, b \in V$, on a :

$$a = a_1 e_1 + a_2 e_2, \quad b = b_1 e_1 + b_2 e_2$$

et :

$$a \wedge b = (a_1 e_1 + a_2 e_2) \wedge (b_1 e_1 + b_2 e_2) = a_1 b_2 e_1 \wedge e_2 + a_2 b_1 e_2 \wedge e_1 = (a_1 b_2 - a_2 b_1) e_1 \wedge e_2$$

(on retrouve donc la définition d'un produit vectoriel dans un espace à deux dimensions)

4.3.4 Le produit géométrique

C'est à partir des notions vues dans les sections précédentes que Clifford inventa le produit géométrique. Clifford était à la recherche d'une algèbre permettant d'étendre le calcul complexe à des espaces de dimensions arbitraires. En prenant pour partie réelle le produit intérieur et pour partie imaginaire le produit extérieur, nous retrouvons l'arithmétique complexe classique, l'idée de Clifford fut de réunir ces deux parties dans une seule et même entité, il définit pour cela le produit géométrique.

Le produit géométrique des vecteurs a et b (écrit simplement ab) est tel que :

$$ab = a \cdot b + a \wedge b.$$

Le produit géométrique possède les caractéristiques suivante

$$\begin{aligned} a(bc) &= (ab)c = abc \text{ (associativité)} \\ a(b+c) &= ab + ac \text{ (distributivité de l'addition)} \end{aligned}$$

Des propriétés des produits extérieurs et intérieurs, on peut déduire que :

$$\begin{aligned} ba &= b \cdot a + b \wedge a = a \cdot b - a \wedge b \\ a \cdot b &= \frac{1}{2}(ab + ba) \text{ et } a \wedge b = \frac{1}{2}(ab - ba) \end{aligned}$$

Nous pouvons donc écrire ces deux produits en fonction du produit géométrique, cela nous permet donc de nous servir du produit géométrique comme base principale et de tout écrire à partir de lui.

4.4 Algèbre géométrique dans \mathbb{R}^3

L'algèbre géométrique dans \mathbb{R}^3 n'est pas l'algèbre la plus efficace pour ce qui est de la représentation géométrique, l'algèbre géométrique conforme (5 dimensions) est bien plus efficace (nous l'étudierons dans le chapitre suivant). Néanmoins pour une bonne compréhension, il est nécessaire de commencer par le commencement.

4.4.1 Algèbre géométrique dans un espace à trois dimensions \mathcal{G}^3

Dans un espace vectoriel de dimension 3 nous pouvons trouver une base de trois vecteurs linéairement indépendant, soit (e_1, e_2, e_3) cette base. avec une telle base, nous pouvons donc former des éléments de Grade 3, on appelle grade d'un objet est le nombre de vecteurs linéairement indépendants le composant.

La base de l'algèbre \mathcal{G}^3 est donc :

$$\begin{array}{ccccccc} 1 & e_i & e_i \wedge e_j & I = e_1 e_2 e_3 & & & \\ 1 \text{ scalaire} & 3 \text{ vecteurs} & 3 \text{ bivecteurs} & 1 \text{ trivecteur (pseudoscalaire)} & & & \end{array}$$

Le pseudoscalaire est un élément intéressant de cette algèbre, puisqu'il a la particularité d'anti-commuter avec tous les vecteurs, nous pouvons donc écrire les bivecteurs de notre bases comme le produit du pseudoscalaire avec un vecteur dual, le dual d'un élément a sera noté a^* .

$$e_1 e_1 = I e_3, e_2 e_3 = I e_1, e_3 e_2 = I e_2$$

Nous venons par conséquent de définir le dual d'un bivecteur, cette notion n'a que peu d'utilité dans \mathcal{G}^3 mais elle permet de faire le rapprochement avec la représentation géométrique usuelle de \mathbb{R}^3 .

En effet nous pouvons avec la notion du dual définir le produit vectoriel de deux vecteurs a et b de \mathbb{R}^3 comme étant le dual associé au produit géométrique de ces deux vecteurs, ceci explique en outre pourquoi le produit vectoriel n'existe pas au delà de \mathbb{R}^3 , au delà de \mathbb{R}^3 le dual d'un bivecteur est au moins de grade 2 (bivecteur, trivecteur,...), objet qui n'a pas de représentation dans la géométrie classique.

4.4.2 Transformations sur les éléments de \mathcal{G}^3

L'un des avantages de la géométrie algébrique est de disposer d'opérations de transformation générale. En ce qui concerne la rotation par exemple, il s'agit de la même opération quelque soit le grade de l'élément de l'algèbre.

Symétrie

Supposons que nous voulions obtenir le symétrique a' d'un vecteur a de \mathcal{G}^3 par rapport au plan orthogonal à un vecteur unitaire m ($m^2 = 1$). Soit $a_{||}$ la composante de a parallèle à m, elle est égale à :

$$\begin{aligned} a_{||} &= a.m \text{ (projection de a sur m)} \\ a_{||} &= a.mm \end{aligned}$$

Soit a_{\perp} la composante de a orthogonale à m, elle est égale à :

$$\begin{aligned} a_{\perp} &= a - a_{||} \\ a_{\perp} &= a \wedge mm \end{aligned}$$

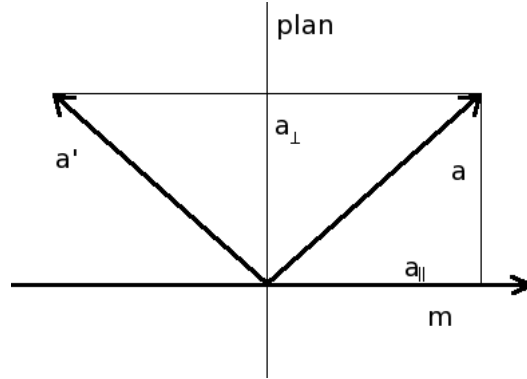


FIG. 4.2 – Exemple de réflexions

Si a' est le symétrique de a alors, il a la même composante orthogonale que a et la composante parallèle opposé à celle de a (voir figure 4.2), a' est donc égal à :

$$a' = a_{\perp} - a_{\parallel} = -mam$$

Il est donc extrêmement aisé de trouver le symétrique d'un vecteur par rapport à un plan avec la géométrie algébrique (cette notion de symétrie se généralise aussi aux bivecteurs et trivecteurs mais n'a pas de sens géométriques dans \mathbb{R}^3).

Rotation

Une rotation dans un plan généré par deux vecteurs unitaires m et n peut-être définie par une des réflexions successives par le plans dont la normale est m , puis par le plan dont la normale est n . Ces réflexions successives sont équivalentes à une rotation de 2θ , ou θ est l'angle orienté entre m et n . A partir de ce constat, nous pouvons définir la rotation a'' d'un vecteur a comme :

$$a'' = -n(-mam)n = nmamn$$

En définissant le rotor $\mathcal{R} = nm$ nous pouvons écrire une rotation ainsi :

$$a \mapsto \mathcal{R}a\tilde{\mathcal{R}} \text{ (ou } \tilde{\mathcal{R}} \text{ signifie } -\mathcal{R}\text{)}$$

Cette notion et cette formule de rotation est généralisable pour n'importe quelle dimension de l'algèbre, n'importe quel grade.

Il serait maintenant intéressant de trouver une formule pour la rotation plus facile à utiliser, en effet pour obtenir une rotation d'angle θ il faut trouver deux vecteurs du plans formant un angle de 2θ , ce n'est évidemment pas pratique.

Pour ce faire nous allons définir \hat{B} comme étant un bivecteur unitaire, pour que \hat{B} soit unitaire il faut que $\hat{B}^2 = -1$.

Intéressons nous maintenant au produit intérieur du bivecteur $m \wedge n$ par lui même, puisqu'il s'agit du produit intérieur le résultat sera forcément un scalaire.

$$\begin{aligned} (n \wedge m).(n \wedge m) &= \langle n \wedge mn \wedge m \rangle^1 \\ (n \wedge m).(n \wedge m) &= \langle nmn \wedge m \rangle \\ (n \wedge m).(n \wedge m) &= n.(m.(n \wedge m)) \\ (n \wedge m).(n \wedge m) &= n.(m \cos(\theta)^2 - n) \\ (n \wedge m).(n \wedge m) &= \cos(\theta)^2 - 1 = -\sin(\theta)^2 \end{aligned}$$

En posant :

$$\hat{\mathcal{B}} = m \wedge n / \sin(\theta)$$

nous obtenons bien :

$$\hat{\mathcal{B}}^2 = -1.$$

Nous pouvons donc écrire \mathcal{R} en terme de bivecteur :

$$\mathcal{R} = \cos(\theta) - \hat{\mathcal{B}} \sin(\theta)$$

ou plus élégamment :

$$\begin{aligned} \mathcal{R} &= e^{-\hat{\mathcal{B}}\theta} \text{ (pour une rotation d'angle } 2\theta) \\ &\quad \text{soit} \\ \mathcal{R} &= e^{-\hat{\mathcal{B}}\theta/2} \text{ (pour une rotation d'angle } \theta) \end{aligned}$$

Nous avons dors et déjà atteint les limites de \mathcal{G}^3 , il nous faut donc si nous souhaitons obtenir des résultats plus intéressants encore, aller voir au delà.

4.5 L'algèbre géométrique conforme (CGA)

4.5.1 Passage à l'espace conforme

L'espace conforme, est un espace vectoriel de dimension 5 et de signature $\mathfrak{U}(4, 1)^2$. Cet espace permet une description de \mathbb{R}^3 , une base de cet espace est la base de \mathbb{R}^3 (e_1, e_2, e_3) auquel on ajoute deux vecteurs orthogonaux à la base de \mathbb{R}^3 et entre eux, e et \bar{e} .

Posons $n = e + \bar{e}$ et $\bar{n} = e - \bar{e}$, le passage de \mathbb{R}^3 au à l'espace conforme \mathbb{E} se fait par l'application F suivante :

$$\begin{aligned} \mathbb{R}^3 &\rightarrow \mathbb{E} \\ x &\mapsto F(x) \\ F(x) &= \frac{1}{2}(x^2 n + 2x - \bar{n}) \end{aligned}$$

¹(A) signifie que l'on ne s'intéresse qu'à la partie scalaire de A

²On appelle (p,q) signature d'un espace vectoriel V, si on peut trouver une base (e_1, \dots, e_{p+q}) de V telle que pour $i \in [1..p]$ $e_i^2 = 1$ et pour $i \in [p+1..n]$ $e_i^2 = -1$

Le passage de \mathbb{E} à \mathbb{R}^3 se fait plus naturellement, il suffit de ne garder que les composantes associées à la base de \mathbb{R}^3 .

La base de l'algèbre (CGA) associée à \mathbb{E} est donc :

$$\begin{array}{cccc}
 1 & e_i & e_i \wedge e_j & e_i \wedge e_j \wedge e_k \\
 1 \text{ scalaire} & 5 \text{ vecteurs} & 10 \text{ bivecteurs} & 10 \text{ trivecteurs} \\
 \\
 e_i \wedge e_j \wedge e_k \wedge e_l & e_1 \wedge e_2 \wedge e_3 \wedge e \wedge \bar{e} & & \\
 5 \text{ 4-vecteurs} & 1 \text{ 5-vecteur (pseudoscalaire)} & &
 \end{array}$$

Nous pouvons rajouter à ces définitions quelques propriétés qui pourraient s'avérer utiles par la suite :

$$F(x).n = -1, \forall x \in \mathbb{R}^3$$

et pour $a, b \in \mathbb{R}^3$ dont les représentations sont $A, B \in \mathbb{E}$:

$$A.B = -\frac{1}{2}(a - b)^2$$

4.5.2 Transformations

Nous savons donc vu dans la partie précédente comment se construisait les rotors de transformation à travers les exemples de la réflexion et de la rotation. Nous allons ici voir d'autres rotors permettant de faire d'autres transformations.

Rotation d'angle θ dans le plan \hat{B} :

rotor :

$$\mathcal{R} = e^{-\hat{B}\theta/2}$$

passage à l'espace conforme :

$$x \mapsto \mathcal{R}x\tilde{\mathcal{R}} \Leftrightarrow F(x) \mapsto F(\mathcal{R}x\tilde{\mathcal{R}})$$

Translation de vecteur a :

rotor :

$$\mathcal{T} = e^{na/2}$$

passage à l'espace conforme :

$$x \mapsto x + a \Leftrightarrow F(x) \mapsto \mathcal{T}F(x)\tilde{\mathcal{T}} = F(x + a)$$

Dilatation de facteur $e^{-\alpha}$

rotor :

$$\mathcal{D} = e^{\alpha e \bar{e} / 2}$$

passage à l'espace conforme :

$$x \mapsto e^{-\alpha} x \Leftrightarrow F(x) \mapsto \mathcal{D}F(x)\tilde{\mathcal{D}} = e^{\alpha} F(e^{-\alpha} x)$$

4.5.3 Les objets primitifs dans la CGA

La CGA permet de définir droites, plans, cercles et sphères comme étant des éléments primitifs, nous n'expliquerons pas ici comment nous aboutissons à ces résultats. Nous nous contenterons juste de les énumérer.

Cercle :

Soit X_1, X_2, X_3 trois points d'un cercle C , on a alors :

$$C = X_1 \wedge X_2 \wedge X_3$$

de rayon

$$\rho = (C^*)^2$$

et de centre

$$C = C^* (1 + \frac{1}{2} C^* n)$$

Sphère :

Soit X_1, X_2, X_3, X_4 quatre points appartenant à la sphère S , alors

$$S = X_1 \wedge X_2 \wedge X_3 \wedge X_4$$

de rayon

$$\rho = (S^*)^2$$

et de centre

$$C = S^* (1 + \frac{1}{2} S^* n)$$

Droite :

Soit X_1, X_2 deux points appartenant à la droite D , alors

$$D = X_1 \wedge X_2 \wedge n$$

de distance à l'origine

$$d = \frac{1}{2} D^* \cdot \bar{n}$$

et de plan normal

$$P_n = D^* - \frac{1}{2} (D^* \cdot \bar{n}) n$$

Plan :

Soit X_1, X_2, X_3 trois points appartenant au plan P, alors

$$P = X_1 \wedge X_2 \wedge X_3 \wedge n$$

de distance à l'origine

$$d = \frac{1}{2}P^* \cdot \bar{n}$$

et de normale

$$n = P^* - \frac{1}{2}(P^* \cdot \bar{n})n$$

De ces définitions nous pouvons voir des similitudes entre le cercle est la droite ainsi qu'entre la sphère et le plan. Ceci s'explique par le fait que dans l'espace conforme l'élément n peut être considéré comme un point à l'infini, par conséquent on peut considérer une droite comme un cercle de rayon infini et un plan comme une sphère de même rayon.³

L'opérateur intersection (meet)

Nous venons déjà de voir le gros potentiel de la CGA, mais l'outil le plus fabuleux que cette algèbre possède est l'opérateur *meet*, \vee .

Cet opérateur unique permet de calculer l'intersection entre n'importe quels éléments de l'algèbre. Cet opérateur est défini comme suivant, soit A et B $\in \mathbb{E}$:

$$A \vee B = (A^* \wedge B^*)^*$$

La seule précaution à prendre concernant cet opérateur, est lors de l'interprétation du résultat, en effet, les résultats donnés par l'opérateur *meet* ne sont pas forcément du grade attendu.

Prenons par exemple S_1 et S_2 deux sphères, on peut s'attendre à ce que leur intersection soit ou un cercle, ou un point ou encore rien du tout, donc un résultat de grade maximum 3, or le résultat donné par $S_1 \vee S_2$ peut être de grade supérieur à 3. Il faut donc alors tout simplement ignorer les composantes de trop grand grade. Une fois ceci fait, il reste à calculer le rayon du cercle d'intersection, si celui-ci est positif, alors l'intersection est un cercle, si il est égal à zéro l'intersection est un point et si le rayon est négatif, alors il n'y a pas d'intersection. On peut pour toujours quelque soit le type d'intersection interpréter le résultat obtenu, mais ces techniques soit simples et intuitives, elle ne seront donc pas décrites ici.

³Ceci peut sembler peu instinctif au premier abord, cependant cela le devient lorsque l'on considère qu'à l'échelle humaine la valeur du rayon du globe terrestre est infini et que nous percevons la surface terrestre comme étant un plan et non une sphère.

Chapitre 5

CONCLUSION

5.1 Travaux accomplis

J'ai donc lors de ce stage réalisé la travail qui m'était demandé, à savoir un moteur haptique pour environnements à six degrés de liberté. Ce moteur permet de réaliser des opérations de manipulations d'objets tridimensionnels à l'intérieur d'environnements composés de 1200 triangles, ce nombre pouvant être grandement augmenté en utilisant un partitionnement de l'espace.

Un des regrets que j'ai à propos de ce moteur haptique, est de n'avoir pas pu faire de comparaisons directes avec les autres moteurs existants, de par la disparités des critères de jugements. Par exemple, si l'on avait à comparer le moteur VPS de McNeely, dont les performances sont mesurées en fonction du nombre de Voxels, à mon moteur, dont les performances sont mesurées en nombres de triangles ¹, il faudrait trouver une relation entre triangles et voxels, et que cette relation respecte une certaine qualité du rendu haptique. Hors il n'existe pas de moyens efficace pour faire ce jugement sur la qualité, les seules méthodes existantes sont la comparaisons des forces générées avec les forces théoriques et les méthodes de jugement de la qualité par un échantillon d'utilisateurs humains. Le problème étant pour la première des deux méthodes que puisque que les forces sont générées en fonction de la distance de pénétration elles sont en générales si éloignés des forces théoriques et que l'on ne peut déterminer lequel des deux moteurs testés est de meilleure qualité, et pour l'autre méthode qu'elle se base sur un facteur humain et par conséquent n'est pas plus fiable que la première méthode. Et pour ce qui est de la comparaison à d'autres moteurs dont les concepts de fonctionnement sont encore plus éloignés de ceux du mien que le VPS, elle me semble tout simplement impossible. Il n'existe de plus pas de benchmark général pour tester les moteurs haptique, ceci rend encore plus impossible toute comparaison directe.

¹En fait les performances de ces deux moteurs sont aussi mesurées en fonction du nombre de PointShells et de SignificantPoints, mais dans l'explication suivante nous pouvons omettre ce détail

Ma recherche d'une théorie permettant de calculer l'intersection de deux polygones quelconque, m'a aussi permis d'augmenter mes connaissances mathématiques, et même si au premier abord cela m'a semblé être une perte de temps, je considère maintenant que cela m'a permis de mieux comprendre beaucoup de chose concernant la géométrie et que par conséquent cela devrait me servir si je continue de faire de la recherche dans le domaine de l'haptique.

5.2 Travaux éventuels à faire

En ce qui concerne le moteur haptique développé durant ce stage, il serait intéressant de voir le bénéfice apporté par le partitionnement de l'espace, je suis d'ailleurs au moment de la rédaction de ce rapport d'implémenter un algorithme de partitionnement basé sur les tables de hachages parfaites ², il ne sera hélas pas terminé assez tôt pour être inclus dans ce rapport.

Il reste aussi à trouver une façon mathématique pour généraliser l'intersection de polygone quelconque, une des pistes possibles sur laquelle je me documente en ce moment est la représentation paramétrique des polygones ³ Ceci nous amène à une autre branche des mathématiques traitant ce genre d'entité, la topologie géométrique (ou topologie différentielle).

5.3 Impressions sur ce stage

Durant ce stage, j'ai notamment pu m'apercevoir de ce qu'était la recherche dans un laboratoire privé, j'ai pu aussi me rendre compte que le facteur humain jouait énormément (notamment dans l'acceptation des articles pour les conférences). J'ai également appris à travailler en équipe, chose dont je n'avais pas l'habitude et partager mes travaux de recherche. J'ai énormément apprécié ce stage, il a renforcé ma conviction de faire de la recherche en informatique mon métier et de poursuivre en thèse l'année prochaine.

²Son gros inconvénient sera de par la nature des tables de hachages parfaites, que au moindre changement de l'environnement il faudra recalculer le partitionnement, ce qui peut être très lourds en temps de calcul

³D'une manière générale tout polygone peut s'écrire sous la forme : $[P_1..P_n] = \{\sum_{i=1}^n \overline{OP_i} \cdot \alpha_i, \sum_{i=1}^n \alpha_i = 1, \alpha_i \geq 0, \forall i\}$. Au quel cas $[P_1..P_n]$ est l'ensemble des points composant le polygone.

Bibliographie

- [1] B. Insko, *Passive Haptics Significantly Enhance Virtual Environments*, PhD thesis, University of North Carolina, 2001.
- [2] Jr. Brooks and al., *Project GROPE - Haptic displays for scientific visualization*, Proceedings of Computer Graphics (SIGGRAPH '90), Vol. 24, pp. 177-185, 1994.
- [3] A. Bejczy and J. K. Salisbury, *Kinematic coupling between operator and remote manipulator*, Advances in Computer Technology Vol. 1, pp. 197-211, 1980.
- [4] M. Minsky, *Computational Haptics : The Sandpaper System for Synthesizing Texture for a Force Feedback Display*, PhD thesis, MIT, 1995.
- [5] T. M. Massie and J. K. Salisbury, *The phantom haptic interface : A device for probing virtual objects*, Proceedings of ASME Haptic Interfaces for Virtual Environment and Teleoperator Systems Vol. 1, pp. 295-301, 1994.
- [6] A. Gregory and al., *H-COLLIDE : A framework for fast and accurate collision detection for haptic interaction*, Proceedings of Virtual Reality Conference, pp. 38-45, 1999.
- [7] J. E. Colgate and J. M. Brown, *Factors affecting the z-width of an haptic display*, IEEE International Conference on Robotics and Automation, pp. 3205-3210, 1994.
- [8] S. J. Lederman R. L. Klatzky and al., *Feeling textures through a probe : Effects of probe and surface geometry and exploratory factors*, Perception and Psychophysics 65(4), pp. 613-631, 2003.
- [9] C. E. Connor and K. O. Johnson, *Neural coding of tactile texture : Comparison of spatial and temporal mechanisms for roughness perception*, Journal of Neurosciences Vol. 12, pp. 3414-3426, 1992.
- [10] R.L. Klatzky and S. J. Lederman, *Perceiving texture through a probe*, Touch in Virtual Environments, ch. 10, pp. 180-193, 2002.
- [11] S. J. Lederman and al., *Perceiving roughness through a probe : Effect of applied force and probe diameter*, Proceeding of the ASME, 2000.
- [12] S. J. Lederman and al., *Perceiving roughness via a rigid stylus : Psychophysical effects of exploration speed and mode of touch*, Haptics-e, 1999.

- [13] M. A. Otaduy and al., *Haptic display of interaction between textured models*, Proceeding of IEEE Visualization, pp. 297-304, 2004.
- [14] C. B. Zilles and J. K. Salisbury, *A Constraint-based God-object Method For Haptic Display*, Proceedings of the International Conference on Intelligent Robots and Systems, Vol. 3., pp. 3146, 1995.
- [15] Krasimi Kolarov Diego C. Ruspini and Oussama Khatib, *The haptic display of complex graphical environnements*, Computer Graphics Proceedings, pp. 345-352, 1997.
- [16] Krasimi Kolarov Diego C. Ruspini and Oussama Khatib, *Haptic interaction in virtual environnements*, Proceedings of the International Conference on Intelligent Robots and Systems, 1997.
- [17] Miguel A. Otaduy and Ming C. Lin, *Sensation Preserving Simplification for Haptic Rendering*, ACM Transactions on Graphics, Vol. 22. pp. 543-553, 2003.
- [18] S. Zhou and R. Klette, *Multiresolution Surface Reconstruction : Edge Collapsing + Simplification Envelopes*, 1997.
- [19] D. Johnson and P. Willemsen, *Six Degree-of-Freedom Haptic Rendering of Complex Polygonal Models*, Proceedings of 2003 Haptics Symposium, 2003.
- [20] P. Willemsen D. Johnson and E. Cohen, *A Haptic System For Virtual Prototyping of Polygonal Models*, Proceedings of Design Engineering Technical Conferences, 2004.
- [21] D. Johnson and E. Cohen, *Spatialized Normal Cone Hierarchies*, Proceedings of ACM Symposium on interactive 3D Graphics, 2001.
- [22] K. Puterbaugh W. McNeely and J. Troy, *Six degree-of-freedom haptic using voxel sampling*, Proceedings of ACM SIGGRAPH, pp. 401-408, 1999.
- [23] M. Renz and al., *Stable Haptic Interaction with Virtual Environments Using an Adapted Voxmap-PointShell Algorithm*, Proceedings of the Eurohaptics Conference, pp. 149-154, 2001.
- [24] Didier Badouel, *An efficient Ray-Polygon intersection*, Graphic Gems, Academic Press, pp.390-393, 1990.
- [25] Tomas Möller and Ben Trumbore, *Fast minimum storage ray-triangle intersection*, Journal on Graphic Tools, Vol. 2, No.1, pp.21-28, 1997.
- [26] F. Velasco and J.C. Torres, *Cells Octree : A New Data Structure for Volume Modelling and Visualisation*, 6th International Fall Workshop on Vision, modeling and visualization, pp. 151-158, 2001.
- [27] Y .Chrysanthou and M. Slater, *Computing Dynamic Changes to BSP trees*, Proceeding of EUROGRAPHICS, pp. 321-332, 1992.
- [28] W. K. Clifford, *On the classification of geometric algebras*, Mathematical Paper, R. Tucker, (Ed.), Macmillan, London, pp. 397-401, 1882.
- [29] D. Hestenes and G. Sobczyk, *Clifford Algebra to Geometric Calculus : A Unified Language for Mathematics and Physics*, Dordrecht, 1984.

- [30] D. Hestenes, *New Foundations for Classical Mechanics*, Dordrecht, 1986.
- [31] D. Hestenes and R. Ziegler, *Projective Geometry with Clifford Algebra*, *Acta Applicandae Mathematicae* 23, pp. 25-63, 1991.