Accélération du calcul d'animations de synthèse

Nicolas Ponsi

ECOLE NATIONALE SUPÉRIEURE
DES MINES DE SAINT-ETIENNE

UNIVERSITÉ JEAN MONNET
SAINT-ETIENNE

No d'ordre: 155ID

THESE

Présentée par Nicolas Ponsi

pour obtenir le titre de

Docteur

DE L'UNIVERSITÉ JEAN MONNET DE SAINT-ETIENNE ET DE
L'ECOLE NATIONALE SUPÉRIEURE DES MINES DE SAINT-ETIENNE

Spécialité Informatique, Synthèse d'images

Accélération du calcul d'animations de synthèse

Soutenue à Saint-Etienne, le 30 janvier 1997

Composition du Jury:

Messieurs	J.P.	Schön	Président
	B.	Arnaldi	Rapporteurs
	R.	Caubet	
	D.	Arques	Examinateurs
	B.	Peroche	

ECOLE NATIONALE SUPÉRIEURE
DES MINES DE SAINT-ETIENNE

UNIVERSITÉ JEAN MONNET
SAINT-ETIENNE

No d'ordre: 155ID

THESE

Présentée par Nicolas Ponsi

pour obtenir le titre de

Docteur

DE L'UNIVERSITÉ JEAN MONNET DE SAINT-ETIENNE ET DE
L'ECOLE NATIONALE SUPÉRIEURE DES MINES DE SAINT-ETIENNE

Spécialité Informatique, Synthèse d'images

Accélération du calcul d'animations de synthèse

Soutenue à Saint-Etienne, le 30 janvier 1997

Composition du Jury:

Messieurs	J.P.	Schön	Président
	B.	Arnaldi	Rapporteurs
	R.	Caubet	
	D.	Arques	Examinateurs
	B.	Peroche	

A mes parents
A Bénédicte

Remerciements

Tous mes remerciements à messieurs Arnaldi et Caubet pour leur lecture attentive de mon manuscrit. Merci à messieurs Arques et Schön pour avoir accepté de participer à mon jury.

Merci à monsieur Peroche pour m'avoir acceuilli dans son laboratoire et pour ses conseils.

Merci à tous les membres du personnel de l'Ecole qui m'ont accordé leur amitié et tout particulièrement à ceux du coin café.

Toute ma reconnaissance à Marc Roelens pour ses très nombreux conseils et pour le dévouement dont il fait preuve. Merci à Bernard Bonnefoy et Florence Dujardin pour leur aide logistique. Merci également à Françoise, Marion et Marie-Line, entre autres pour m'avoir libéré de tous soucis le jour de ma soutenance.

Merci enfin à mes amis, parents, futurs beaux-parents et surtout à Bénédicte, ma future épouse, pour m'avoir soutenu pendant la durée de ma thèse

Table des matières

Introduction

Fondés pour la plupart sur [WITH80], les logiciels de synthèse qui utilisent la technique du lancé de rayons sont très répandus, grâce à la relative simplicité de celle-ci et à la qualité de ses résultats. Cet algorithme nécessite cependant le calcul de nombreuses intersections entre les droites représentant les rayons lumineux et la description géométrique des objets à visualiser, qui peut être complexe. Dans sa version initiale. les temps de calculs induits sont en conséquence très importants.

Ainsi de nombreux auteurs ont tenté de réduire le nombre d'intersections à calculer pour la production d'une image de synthèse. Cette réduction utilise différents types de *cohérences* présents dans la création d'une image de synthèse, c'est à dire le fait que certains éléments sont localement constants. Nous exposerons ces travaux dans notre premier chapitre en tentant de les classer selon le principe de leur idée fondatrice. On fera ainsi apparaître les relations qui existent entre eux et les différentes orientations principales choisies.

Nous présenterons dans un second temps une méthode qui permet d'évaluer l'impact du mouvement relatif des objets sur leur apparence dans l'image produite. Tous

les mouvements seront pris en compte: les translations aussi bien que les rotations. Il s'agira, à partir des mouvements dans l'espace à la fois de l'observateur et des objets, de fournir une information fiable sur les modifications géométriques de la projection de ces derniers entre deux instants de l'animation.

Cette information nous permettra ensuite de proposer un algorithme pour l'accélération de la production d'animations de synthèse. On pourra en effet détecter les objets dont l'apparence n'est pas modifiée entre deux instants de l'animation. On évitera ainsi les calculs inutiles en ne traitant à chaque image que les objets qui ont réellement besoin de l'être.

Cet algorithme aura pour objectif de produire des animations visuellement correctes le plus rapidement possible, sans pour autant être limité dans les mouvement ni de l'observateur, ni des objets. Il devra de plus pouvoir être utilisé dans les conditions du temps-réel, où l'on ne connaît pas l'ensemble de l'animation a priori. Les seules informations nécessaires seront les vitesses initiales des différents objets et de l'observateur, cela afin de réaliser une prédiction de leurs mouvements futurs.

Nous conclurons enfin sur les performances et limitations de notre méthode d'accélération ainsi que sur les développements ultérieurs qui peuvent en être envisagés.

CHAPITRE I

De la cohérence avant tout!

I-1) Introduction

De nombreux auteurs ont tenté de réduire le nombre d'intersections à calculer pour la production d'une image de synthèse. Cette réduction utilise différents types de *cohérences* présents lors de la création d'une image de synthèse, c'est à dire le fait que certains éléments sont localement constants.

Les techniques les plus fréquemment utilisées utilisent les cohérences internes aux différentes structures produites lors de la synthèse d'une image: les objets de la scène, l'arbre des rayons et l'image produite.

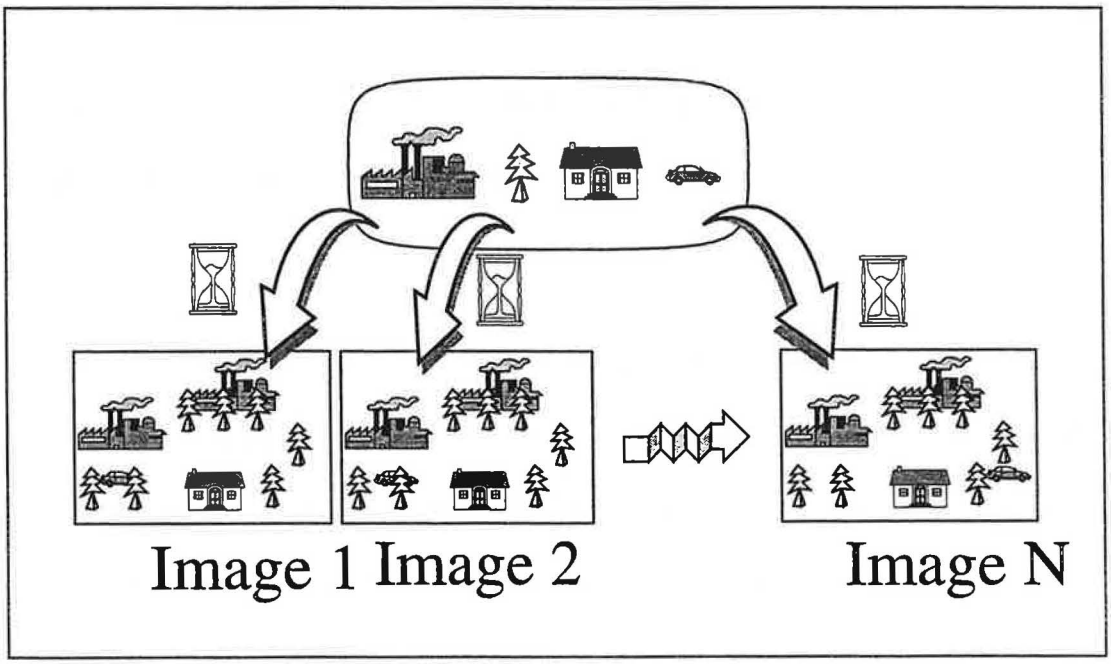
La matière n'est pas uniformément répartie dans une scène: elle est regroupée en objets qui sont souvent eux-mêmes réunis. C'est ce que l'on peut appeler *la cohérence spatiale de l'espace des objets*. En une région de l'espace, il est en conséquence inutile de tester une intersection précise avec l'ensemble des objets de la scène. Les méthodes issues de ce type de cohérence ont pour principe de subdiviser l'espace en éléments

simples. On peut ainsi rapidement savoir quels sont les objets susceptibles d'être rencontrés.

Une image n'est également pas aléatoire. Il est visible que la répartition des couleurs crée des zones d'homogénéité. Les méthodes utilisant cette c*ohérence spatiale de l'image* tenteront en conséquence de détecter ces homogénéités afin de n'avoir à faire qu'un unique calcul de couleur sur celles-ci.

La *cohérence de l'espace des rayons* utilise une conséquence des techniques de lancer de rayons. De nombreux rayons sont en effets similaires: ils ont à peu près la même origine et la même direction. Cette remarque est principalement valable pour les rayons primaires qui sont tous issus de l'oeil et les rayons d'ombres qui convergent vers une source lumineuse. Des calculs équivalents sont donc réalisés pour des rayons similaires. Il parait en conséquence intéressant de regrouper les calculs de ces rayons.

Figure 1. Le calcul classique d'une animation

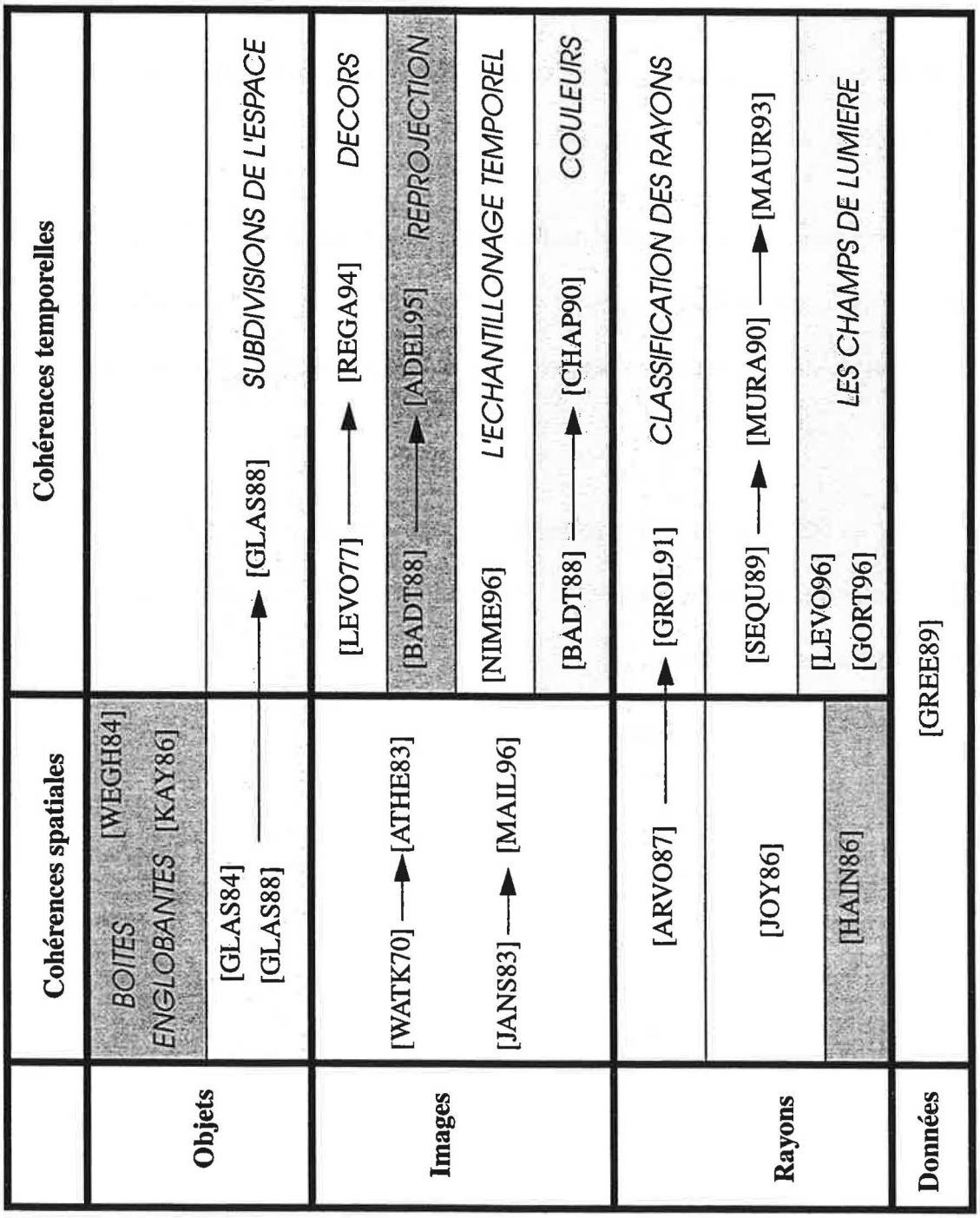


Classiquement, le calcul d'une animation est vue comme les calculs successifs d'une suite d'images. Cependant, comme dans la Figure 1, page 8, le mouvement des objets est généralement réaliste donc continu, les scènes à visualiser varient peu d'une image à l'autre. Le calcul d'une animation peut donc être vue comme les représentations d'une même scène à des instants successifs. Les différentes structures nécessaires à la synthèse des images sont en conséquences elles aussi semblables d'une image à l'autre. C'est la notion de *cohérence temporelle*.

Enfin, la notion de *cohérence des données* concerne la répartition des données utiles au logiciel de lancer de rayons dans les différents types de mémoires présentes dans un ordinateur. Il s'agit de gérer les mémoires afin de limiter les temps d'accès aux données.

L'ensemble de ces types de cohérences est regroupé dans la Figure 2, page 10 ainsi que des articles qui ont utilisé ces notions. On détaillera dans la suite de ce chapitre les utilisations qui en ont été faites.

Figure 2. **Les utilisations de la cohérence**



I-2) Les Cohérences spatiales

I-2-1) La cohérence des objets

Ces méthodes ont pour hypothèse qu'il est préférable de calculer un grand nombre d'intersections avec des objets dont la géométrie est simple (sphère, cube ou encore divers parallélépipèdes) qu'un nombre certes moins important d'intersections, mais avec des objets complexes.

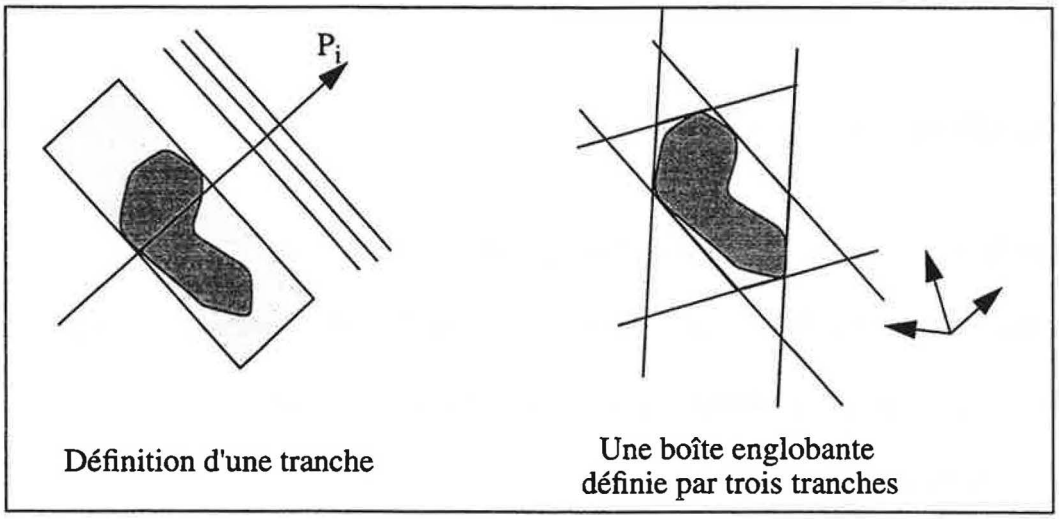
I-2-1-1) Les boîtes englobantes [WEGH84][KAY86]

La technique des volumes englobants organise une hiérarchie de volumes simples contenant les primitives complexes. Ainsi, un rayon ne peut intersecter une primitive que s'il intersecte un volume englobant qui le contient.

Le problème du choix de la forme des boîtes englobantes est crucial comme l'ont exposé *Weghorst et al.* dans [WEGH84]. Il faut tout en effet considérer deux paramètres qui peuvent être antagonistes. Le coût de l'intersection entre la boîte et un rayon doit être faible, ce qui implique des primitives simples. D'un autre côté, il faut limiter la possibilité qu'un rayon coupe la boîte sans couper l'objet qu'elle contient: la primitive doit approcher le mieux possible la forme du contenu.

Un compromis aujourd'hui accepté est celui des «slabs» proposés par *Kay et al.* [KAY86]. Le terme «slab» peut être traduit par «tranche». Il s'agit en effet d'entourer l'objet par des tranches d'espace définies par des plan parallèles. Ces plans sont définis par leur vecteur normal.

Figure 3. **Les boîtes englobantes définies par des tranches**



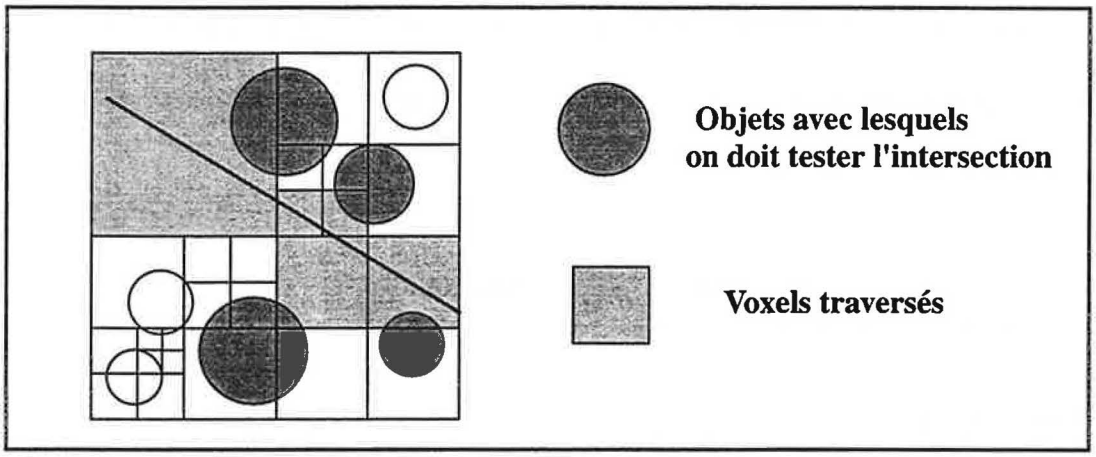
Le calcul des tranches est aisé si l'objet est un polyèdre. Il est alors défini par un ensemble de points. Il est facile de projeter ces points sur le vecteur normal. Les points projetés extrémaux définissent alors les positions des plans de la tranche. Le cas des surfaces implicites, comme les sphères, est résolu également par la recherche des extrema des projections, au prix d'un problème de minimisation.

I-2-1-2) Les subdivisions hiérarchiques de l'espace [GLASS84] [GLAS88]

L'arbre octal ou octree est une subdivision hiérarchique non-uniforme de l'espace 3D. *Glassner* en propose dans [GLASS84] une version adaptée à la synthèse d'images. On divise de manière récursive un cube contenant l'ensemble de la scène. A chaque étape, un cube «père» peut être scindé en ses huit cubes «fils». Par analogie avec les pixels d'une image, ces cubes sont nommés «voxels». A chaque voxel est associée la liste des objets susceptibles d'être coupés par un rayon le traversant. On considère qu'un objet est candidat s'il coupe une des faces du cube ou s'il est inclus dans celui-ci. La division s'arrête lorsque le nombre d'objets candidats est inférieur à un certain seuil où

si l'on atteint une taille limite. L'avantage de cette structure est le non-recouvrement des éléments de subdivision, contrairement aux boîtes englobantes.

Figure 4. Un exemple d'octree



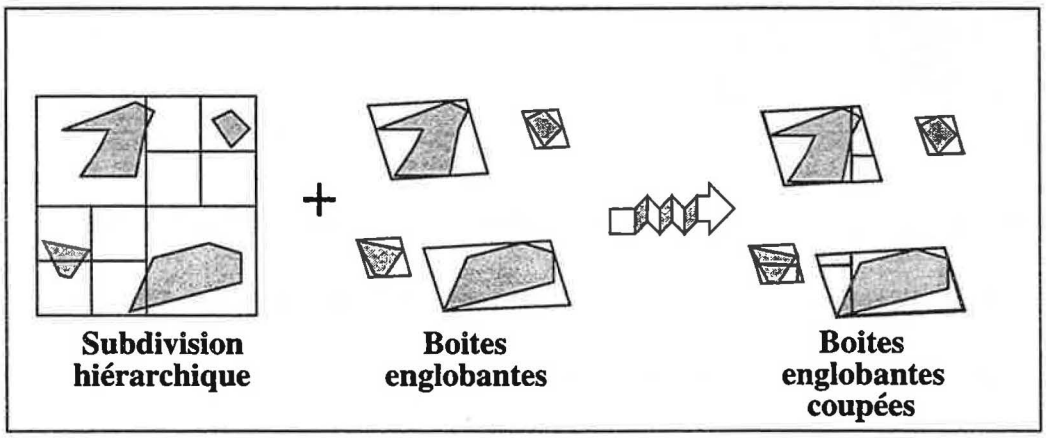
L'octree est classiquement stocké sous la forme de l'arborescence qui le représente. *Glassner* donne à chaque voxel un nom qui lui sert de codage dans une table de hachage. Le cube englobant la scène porte a pour nom «1». Les fils d'un voxel sont ensuite nommés en multipliant le nom du père par dix et en y additionnant leur numéro, de un à huit. Les enfants de «1534» sont par exemple «15341», «15342»,..., «15348». On peut ainsi très rapidement parcourir l'arbre et retrouver les objets candidats associés à chaque voxel par la table de hachage.

Lors du parcours des voxels rencontrés par un rayon, le déplacement d'un voxel dans son voisin est réalisé en calculant un point de l'intérieur du voisin. On commence par calculer le point de sortie du voxel précédent. On se déplace ensuite sur le rayon de la moitié de la dimension du voxel le plus petit de l'octree.

Cette méthode permet de construire une subdivision de l'espace adaptée à la répartition des objets qui s'y trouvent. Elle implique cependant un parcours de nombreux voxels, même pour un rayon qui ne coupe aucun objet comme dans la Figure 4,

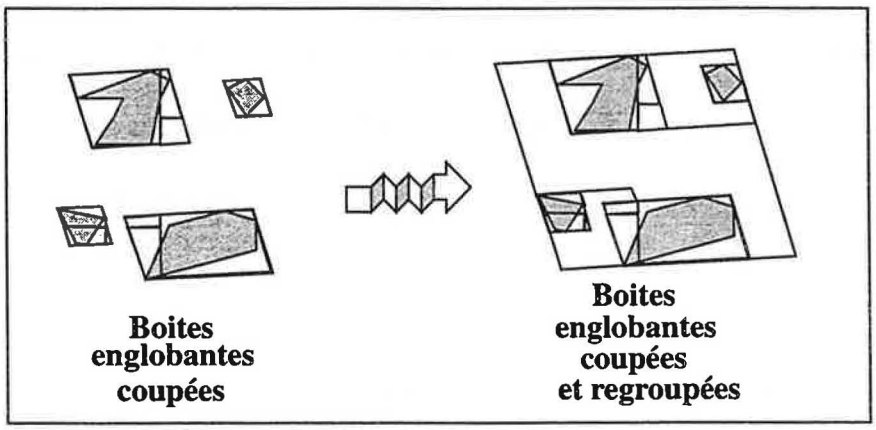
page 13. *Glassner* propose dans [GLAS88] de réduire ce problème en adjoignant à l'octree une hiérarchie de boîtes englobantes. A la scène est associé un octree. A chaque objet est de plus associée une boîte englobante, qui peut être une tranche. Dans la Figure 6, page 14, nous prendrons seulement deux tranches pour plus de simplicité. On réalise ensuite la coupe (ou clipping) des boîtes englobantes par la subdivision hiérarchique.

Figure 5. **La coupe des boîtes englobantes**



On regroupe ensuite ces boîtes englobantes coupées dans de boîtes plus grandes en respectant la hiérarchie de la subdivision:

Figure 6. **Le regroupement des boîtes englobantes**



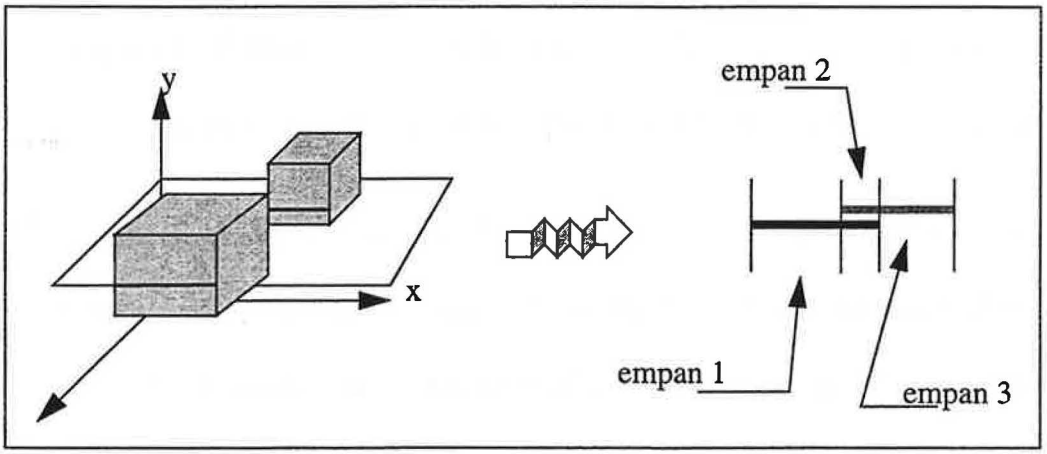
On obtient ainsi une hiérarchie de boîtes englobantes qui possède les avantages des boîtes classiques et des subdivisions hiérarchiques: des englobants plus fins et qui ne se recouvrent pas.

I-2-2) La cohérence de l'image

I-2-2-1) L'algorithme de balayage [WATK70][ATHE83]

L'algorithme de balayage simplifie le problème du calcul d'une image, à deux dimensions, par les calculs de multiples droites à une dimension. Il réalise en effet un balayage vertical de l'image sur des objets polygonaux.

Figure 7. L'algorithme de balayage



On obtient ainsi une liste de segments qui permet de construire les «empans», c'est à dire les portions de la droite de balayage comprises entre deux extrémités de segments. Pour un empan, si les segments ont le même ordre sur leurs profondeurs à leurs extrémités, alors le segment est tracé. Sinon, c'est que les segments se coupent et on les subdivise.

Cet algorithme utilise la cohérence de l'image d'objets polygonaux dans le cas de la projection orthogonale. Il a été étendue par la suite aux scènes CSG facétisées dans [ATHE83] en résolvant les opérations booléennes en chaque empan.

I-2-2-2) L'homogénéité des couleurs [JANS83][MAIL96]

Une image contient très souvent des zones d'homogénéité, c'est à dire des zones où les couleurs varient «peu».

Dans [JANS83], cette propriété est utilisée en divisant récursivement l'image afin de détecter ces zones homogènes. On commence par la subdiviser en blocs de taille fixée. Chaque bloc est à son tour divisé en quatre sous-blocs. La couleur du coin inférieur-gauche de chaque sous-blocs est alors comparée à celle de ses voisins. Si une de ces différences est supérieure à un seuil, alors la subdivision est répétée. La récursion s'arrête soit sur une zone homogène, soit si le bloc a atteint une taille minimale.

Cet algorithme se heurte à plusieurs problèmes. Premièrement, *Jansen* ne précise pas quelle est la représentation des couleurs qu'il utilise, quelle est la distance qu'il applique sur celle-ci et comment fixer le seuil d'homogénéité. L'utilisation de blocs carrés implique de plus une détection uniquement des zones d'homogénéité carrées. Le critère retenu pour l'homogénéité peut enfin se révéler faux. Il peut en effet exister des variations de couleurs à l'intérieur d'une zone, même si les couleurs des coins sont équivalentes.

Afin de répondre à ces problèmes, *Jean-Luc Maillot* propose sa méthode de synthèse d'images par *progressivité* dans [MAIL96]. Comme précédemment, cette technique permet en effet d'obtenir l'image par raffinements successifs d'une première image grossière.

L'algorithme commence par un échantillonnage aléatoire du plan de l'image organisé au sein d'un quadtree. Le développement de cet arbre est ensuite tributaire de l'homogénéité ou non de la feuille terminale: si l'homogénéité n'est pas vérifiée, alors une subdivision de la feuille est réalisée. De plus, chaque région de l'arbre doit contenir un nombre minimal N de points afin d'avoir un test statistique fiable. Après de nombreuses mesures sur un panel d'images test, *Jean-Luc Maillot* propose les minima suivants:

Taille	**2x2**	**4x4**	**8x8**	**16x16**	**>=32x32**
N minimal	3	12	39	66	71

Pour être homogène, une zone de l'image doit répondre à deux critères. Le premier est une cohérence spatiale: tous les rayons doivent couper le même objet. Le second est que les couleurs des points se trouvent avec un certain risque dans une même sphère unitaire de l'espace L*u*v. On considère en effet que cet espace correspond aux sensations psycho-visuelles du système visuel humain.

Selon la complexité de la scène, les gains de temps obtenus varient entre 60 et 5%. La proportion de rayons générés varie entre 30 et 88% selon les tests effectués. Les résultats obtenus sont de bonne qualité même si certaines inhomogénéités très locales peuvent ne pas être détectées. L'auteur propose d'étendre le test de cohérence spatiale à la détection de zones d'ombre ou de reflets.

Cette méthode a de plus l'avantage de fournir très rapidement des images intermédiaires, qui permettent de valider la description de la scène que l'on cherche à visualiser.

I-2-3) La cohérence de l'arbre des rayons

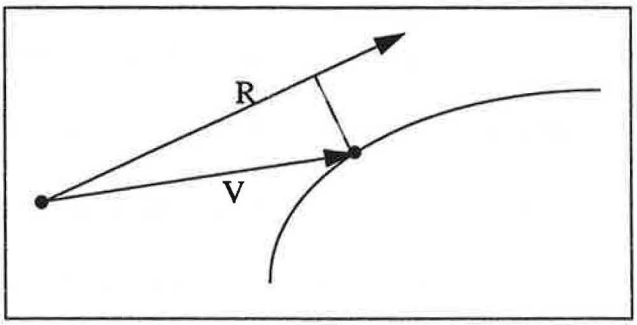
I-2-3-1) Une première classification des rayons [JOY86]

Les méthodes de classification des rayons tentent de regrouper les rayons en classes homogènes afin de tirer profit de leur cohérence.

Joy et al. dans [JOY86] proposent d'utiliser un calcul d'intersection déjà réalisé afin d'accélérer celui d'un rayon «proche», chaque intersection étant calculée par une méthode de quasi-newton.

Soit une surface paramétrique S(u,v) {x(u,v), y(u,v), z(u,v)} et le rayon défini par son origine P(xp, yp, zp) et sa direction normalisée R(xr, yr, zr). Le carré de la distance d'un point de la surface au rayon est $F(u,v) = |V|^2 - (V.R)^2$ avec V = S(u,v) - P.

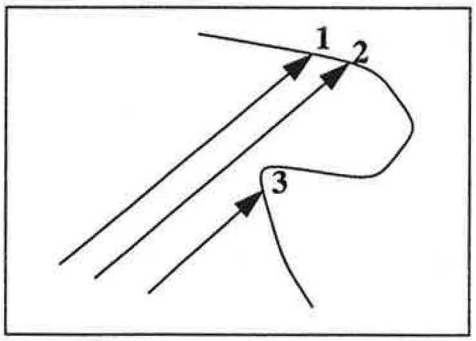
Figure 8. Distance d'un point de la surface au rayon



Les points d'intersection sont ceux pour lesquels F(u,v)=0. Comme beaucoup d'autres techniques, la méthode de quasi-newton converge vers la solution à partir d'un point de départ que l'on doit prendre le plus proche possible de la solution. Il faudrait en conséquence connaître la solution au problème pour pouvoir tenter de le résoudre de manière efficace! Il parait donc logique d'utiliser une intersection calculée comme point de départ pour la recherche de la suivante. Ce processus n'est cependant valable que

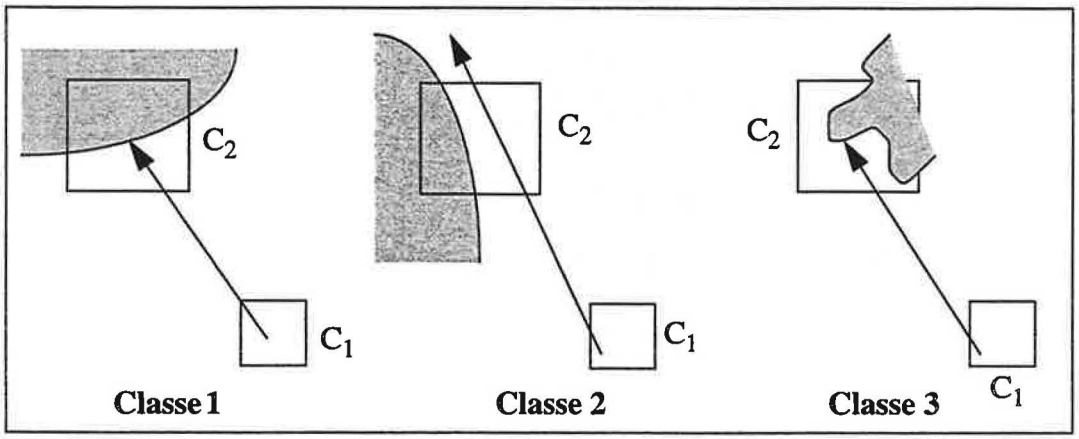
dans certains cas selon la cohérence des rayons. Cette cohérence est traitée du point de vue des intersections. Ainsi, dans la figure ci-dessous, il y a cohérence entre 1 et 2, mais pas entre 2 et 3.

Figure 9. **La cohérence des rayons**



Afin de détecter ces problèmes, *Joy et al.* découpent l'espace sous la forme d'un octree [GLASS84] et répartissent les relations entre cellules en trois classes:

Figure 10. **La classification des cellules**



- Classe 1: Pour tout rayon issu de C1 passant par C2, F(u,v) a un minimum unique: la méthode de cohérence peut être employée.
- Classe 2: Pour certains rayons, il existe un minimum unique supérieur à zéro. Pour d'autres, il existe deux minima égaux à zéro. La méthode de cohérence peut être employée mais il faut vérifier que l'on trouve bien la bonne solution.
- Classe 3: celles qui ne sont ni de classe 1, ni de classe 2. La méthode de cohérence ne peut pas être employée.

Dans les exemples choisis par les auteurs, la convergence est assurée dans 99% des cas en moins de trois itérations mais celui-ci reconnaît lui-même que la classifica-

tion est très lente, ce qui relativise l'accélération espérée. Aucun résultat sur les temps de calcul n'est d'ailleurs fourni.

I-2-3-2) Une partition de l'espace [ARVO87]

Arvo et al. [ARVO87] se penchent sur une *subdivision de l'espace dirigée par une autre classification des rayons*. En effet, ils remarquent que les subdivisions dirigées par les objets de la scène du type octree [GLASS84] imposent le parcours de nombreux éléments même à un rayon qui ne coupe aucun objet.

Ils se placent dans l'espace des rayons, qui comporte cinq dimensions: trois pour la position de l'origine et deux pour la direction. Les auteurs préconisent d'utiliser pour la direction la norme infinie comme ci-dessous:

$$w = \frac{d}{\|d\|_\infty} = \frac{(d_x, d_y, d_z)}{\mathrm{Max}(|d_x|, |d_y|, |d_z|)}$$

$$(u, v) = \begin{cases} (w_y, w_z)(\mathrm{si} w_x = \pm 1)\,\mathrm{sinon} \\ (w_x, w_z)(\mathrm{si} w_y = \pm 1)\,\mathrm{sinon} \\ (w_x, w_y)(\mathrm{si} w_z = \pm 1) \end{cases}$$

On a ainsi une bijection entre [-1,1]x[-1,1] et les rayons traversant chaque face du cube unitaire. On associera chaque face du cube avec son axe dominant (+X, -X, +Y, -Y, +Z ou -Z) afin de référencer chaque zone de l'espace ou faisceau.

Ils divisent ensuite cet espace en sous-espaces E1, E2,..., Em auxquels sont associés les objets potentiellement intersectés C1, C2,..., Cm. Le but de la subdivision est

que chaque classe Ci soit suffisamment petite pour que tous les rayons de Ei rencontrent les mêmes objets.

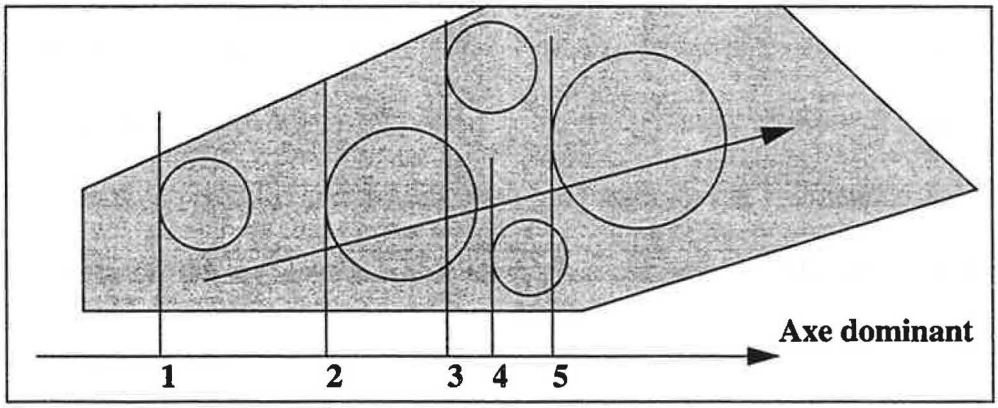
Cette subdivision débute par la recherche d'un sous-espace E contenant tous les rayons. Une solution proposée est de prendre B x [-1,1] x [-1,1] où B est une boite englobante des objets formée par des plans perpendiculaires aux axes. Si l'observateur n'est pas inclus dans B, alors on déplace l'origine du rayon jusqu'à son intersection avec B. On obtient ainsi rapidement des résultats pour des rayons qui ne coupent pas B mais on peut perdre une partie de la cohérence des rayons primaires qui ont tous la même origine.

Chacun des six sous-espaces de E est ensuite subdivisé alternativement en deux selon chacun des cinq axes restants. Cette subdivision est réalisée de manière paresseuse, c'est à dire uniquement lorsque l'on rencontre un rayon qui se trouve dans une région qui peut encore être découpée. Les auteurs proposent deux heuristiques pour tester la fin de la subdivision: on arrête soit quand le nombre d'objets de Ci est inférieur à un certain seuil, soit quand la taille du sous-espace est suffisamment petite.

Afin de classifier les objets, c'est à dire de tester quels sont les objets potentiellement intersectés, deux méthodes sont retenues par les auteurs comme offrant un bon compromis entre la pertinence et la rapidité du test. La première utilise la propriété que si un plan sépare le faisceau et l'objet, alors on peut rejeter ce dernier. Le test est alors réalisé avec les quatre plans qui s'appuient sur le faisceau et en remplaçant l'objet par une enveloppe convexe. La seconde méthode est le calcul d'intersection cône-sphère: chaque objet est remplacé par sa sphère englobante, chaque faisceau par un cône.

Dans le but d'optimiser la recherche d'intersections, les objets sont triés dans chaque faisceau selon leur abscisse sur leur axe dominant. Sur la figure ci-dessous, seuls les deux premiers objets doivent en conséquence être testés.

Figure 11. **Le tri intra-faisceau**



Cette méthode permet une accélération importante des calculs, due à la fois à la partition de l'espace dirigée par les rayons et au tri interne à chaque sous-espace. Par rapport à l'octree de [GLASS84], les auteurs revendiquent une accélération de l'ordre de dix, cette accélération provenant d'une meilleure décomposition de l'espace, au moins pour les rayons primaires, qui ont tous la même origine et pour les rayons d'ombre, qui ont tous la même destination. Le principe parait moins intéressant pour les rayons réfléchis et réfractés.

I-2-3-3) Le tampon de lumière ou «light buffer» [HAIN86]

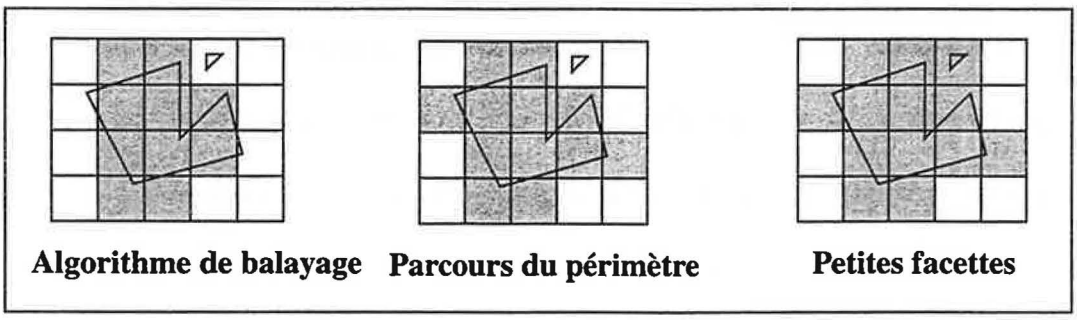
Cette méthode a pour objectif de réduire les temps de calculs liés à l'ombrage. Elle préconise un pré-traitement des rayons d'ombre pour construire la structure désignée sous le nom de tampon de lumière.

Cette structure est en fait un cube, centré sur chaque source lumineuse, aligné par rapport aux axes. Chaque face de ce cube est discrétisée en cellules qui contiennent une

liste décrivant les facettes visibles depuis la source à travers chacun d'elles. Pour chaque facette, on conserve le numéro de l'objet visible, celui de la facette et sa distance à la source. Il est également possible de remplacer un objet-non facétisé par un volume englobant facétisé et en prenant quelques précautions au cours des tests.

Les informations des cellules sont obtenues grâce à un algorithme de balayage que l'on effectue avec la source comme observateur et la face comme écran de projection. Afin de ne pas oublier de morceaux de facettes, on parcourt ensuite le périmètre de chacune d'entre-elles. De plus, on vérifie que toutes ont été marquées au moins une fois afin de détecter celles qui sont plus petites qu'un élément de discrétisation. La distance de la facette associée à chaque cellule est le minimum des distances sur celle-ci. Il est également possible de marquer de manière spéciale les faces qui recouvrent toute une cellule: tout objet qui se trouve derrière est en effet masqué sans qu'un test d'intersection soit nécessaire.

Figure 12. La détection des facettes



Lorsque l'on lance un rayon d'ombrage vers la source, on détecte par quelle cellule il passe. On teste ensuite chaque facette associée à cette cellule en commençant par les plus proches de la source.

Cet algorithme obtient des facteurs d'accélération variant de 4 à 30 dans les exemples proposés par les auteurs. Il faut noter qu'il peut également s'appliquer aux

animations où ni les sources lumineuses ni les objets ne sont mobiles. La taille de la discrétisation est cependant difficile à fixer a priori puisqu'elle dépend de la scène. Une maille trop petite demandera en effet beaucoup de travail tandis qu'une grosse maille n'utilisera pas toute la cohérence des rayons. Il ne s'applique de plus qu'aux rayons d'ombrage et de manière directe qu'aux scènes facétisées.

I-2-4) La cohérence des données [GREE89]

On suppose habituellement que l'ensemble des données nécessaires à la synthèse d'une image peut être stockée dans une mémoire vive infinie. Dans la pratique, avec la complexité croissante des scènes, c'est une mémoire virtuelle qui gère les transferts entre différents type de mémoires vives et une mémoire disque, qui peut être considérée comme infinie.

Green et Paddon remarquent cependant sur quelques exemples une certaine cohérence des données. Cette cohérence est le fait que l'on accède à certaines données plus fréquemment qu'à d'autres. Ils proposent en conséquence de répartir les données dans différents types de mémoires, de temps d'accès plus ou moins importants. Leur répartition se fait tout d'abord entre la mémoire cache du processeur et le reste de la mémoire vive. La mémoire cache est ensuite répartie en une zone de données statiques et une zone d'échange avec le reste de la mémoire vive.

Leur implantation est faite dans le contexte d'une mémoire distribuée sur plusieurs processeurs. La cohérence est ici augmentée par le fait que chaque processeur gère une zone carrée de l'image. Dans le cas de l'utilisation d'un octree [GLAS84], leurs mesures expérimentales montrent qu'une proportion de 7 pour 3 est optimale pour la répartition entre la description des objets et celle des voxels de la mémoire cache du

processeur. La répartition à l'intérieur de la mémoire cache dépend principalement de la taille de celle-ci. Des courbes expérimentales permettent de définir cette répartition.

Afin de déterminer les données les plus accédées, ils calculent une image à faible résolution. Une image 32x32 est suffisante dans leurs exemples pour définir une bonne approximation de la répartition des données pour une image 512x512.

Cette idée de cohérence des données est originale et n'est pas naturelle à la communauté de la synthèse d'images, qui a l'habitude de pré-supposer une mémoire virtuelle infinie et homogène. Elle permet cependant des accélérations de l'ordre de deux sur les exemples proposés et peut donc être en envisagée comme une étape d'optimisation de l'implémentation d'un algorithme.

I-3) Les cohérences temporelles

I-3-1) Le format MPEG

MPEG est l'acronyme de Motion Pictures Experts Group. Il s'agit en effet d'une tentative de création d'un standard pour la compression de données à la fois vidéo et audio. Nous ne nous intéresserons ici qu'à l'aspect vidéo de compression d'une succession d'images connues à l'avance. Il s'appuie sur les techniques standardisées sous les noms JPEG [FUHR95a], pX64 ou H.261 [LIOU91] qui utilisent le principe de cohérence à la fois interne à une image et externe, c'est à dire entre les différentes images d'une animation. L'idée importante de ce type d'algorithme est qu'il découpe l'image en zones rectangulaires ou blocs qu'il essaye ensuite de comparer dans le domaine des fré-

quences. On peut ainsi réaliser un codage interne en recherchant les blocs équivalents, c'est à dire dont la différence est inférieure à un certain seul. On peut également réaliser un codage externe en comparant les blocs de deux images distinctes. Le pari de la compression MPEG est celui de la cohérence temporelle: entre deux images peu éloignées dans le temps, peu de blocs sont modifiés.

Chaque image de l'animation peut être compressée, en utilisant sa cohérence interne, par un algorithme de compression à base de DCT (Discrete Cosinus Transformation) comparable au format JPEG. Nous ne traiterons pas cette compression ici puisque notre intérêt porte sur la cohérence entre images. Le lecteur pourra se rapporter à [FURH95a] pour plus de détails. La cohérence externe utilise le même principe en recherchant les blocs équivalents entre plusieurs images de l'animation. Une fois la compression MPEG réalisée, les images peuvent être séparées en trois catégories utilisant trois algorithmes différents.

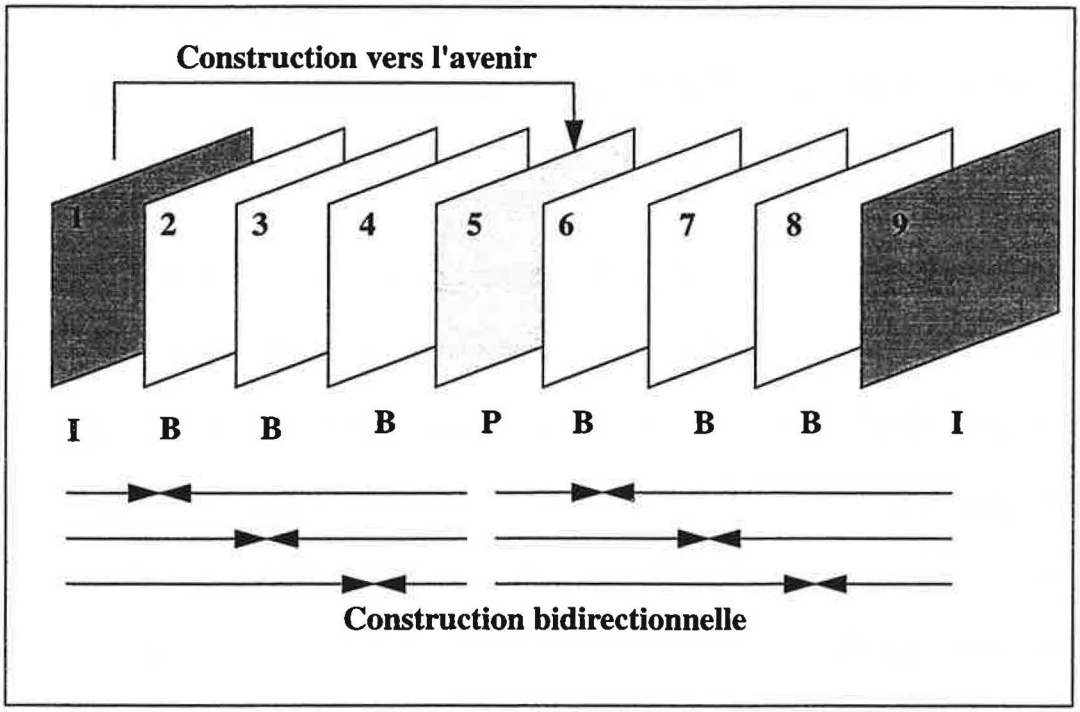
Les images de type I pour image interne sont indépendantes et contiennent toute l'information nécessaire à leur affichage. Elles sont uniquement la compression au format JPEG des images initiales et donnent la borne inférieure de compression de MPEG.

Les images de type P sont des images prédites à partir d'images précédentes de type I ou P en recherchant des blocs identiques, au seuil de qualité fixé près, dans ces images antérieures. Si aucun bloc d'une image antérieure («venant du passé») ne peut être considéré comme valide, alors, comme dans les images I, un codage interne est réalisé pour ce bloc.

Les images de type B sont des images bidirectionnelles, c'est à dire qu'elles sont obtenues à partir de blocs de deux images, l'une P et la seconde I, l'une venant du passé

et l'autre du futur de l'image à reconstruire. Un bloc d'une image B peut être obtenu de trois façons différentes: il peut provenir d'une image du passé, de l'avenir ou être une combinaison d'un bloc du passé et d'un bloc de l'avenir. Comme pour les images de type P, si aucune de ces trois possibilités ne donne un résultat satisfaisant, un codage interne est toujours possible.

Figure 13. **Les différents types d'images**



D'après la figure ci-dessus, il est bien entendu nécessaire de connaître les images 1 et 5 pour construire les images intermédiaires. Ainsi, lors de la phase de visualisation de l'animation, on fournira les informations permettant de reconstruire les images dans l'ordre {1,5,2,3,4,9,6,7,8}.

Des informations sur MPEG peuvent également être trouvées sur le serveur http://www.mpeg.org/. L'organisation y revendique des compressions de l'ordre de 100, ce qui met bien en évidence l'intérêt de la cohérence temporelle. Ceci nous incite à étudier l'usage qui en a été fait dans la production d'animations de synthèse. On regrou-

pera les algorithmes employés sous les appellations de cohérence des objets, des images ou enfin des arbres des rayons.

I-3-2) La cohérence des objets

I-3-2-1) Une subdivision hiérarchique de l'espace 4D [GLAS88]

Glassner propose d'étendre à la dimension temporelle la subdivision hiérarchique de l'espace en boîtes englobantes (Voir *Les subdivisions hiérarchiques de l'espace [GLASS84] [GLAS88]*, page 12.). Il se place alors dans un espace à quatre dimensions.

Dans l'espace 3D, il existe trois axes et huit octants. Sept plans sont alors naturels: trois plans principaux sont perpendiculaires aux axes et quatre plans auxiliaires coupent deux octants chacun selon la diagonale. Dans l'espace à quatre dimensions, on arrive de même à douze plans (quatre plus huit). *Glassner* propose donc des boîtes englobantes formées des douze tranches correspondantes. La subdivision hiérarchique est elle aussi étendue au domaine temporel.

Les rayons temporels sont représentés par une paire d'éléments à quatre dimension, un pour l'origine, l'autre pour la direction. Si l'on suppose la vitesse de la lumière infinie, la composante temporelle de la direction est 0.

La subdivision de l'espace 4D est réalisée comme un pré-traitement. Il est ensuite simple de lancer un rayon temporel dans la subdivision de l'espace 4D. Les gains espérés sont du même ordre que ceux obtenus avec la subdivision hiérarchique de l'espace 3D en boîtes englobantes. Le nombre de tests d'intersection est réduit mais on n'utilise pas complètement la cohérence temporelle. En effet, l'apparence d'un objet, même statique, doit être recalculée pour chaque image.

I-3-3) La cohérence de l'image

La cohérence temporelle de l'image peut être définie comme la conservation de certaines propriétés de l'image au cours du temps. Dans toutes les méthodes utilisant cette propriété, on se sert en conséquence d'une image calculée de manière classique pour en déduire ses suivantes.

I-3-3-1) Les algorithmes à décor [LEVO77] [REGA94]

Cette idée est très proche du dessin d'animation manuel où la superposition de feuilles transparentes ou celluloïds permet l'utilisation du même décor tout au long d'une séquence. Ainsi, dès 1977, *Marc Levoy* [LEVO77] s'inspire-t-il d'une machine mise au point par les studios Disney dans les années 30 pour proposer un algorithme de synthèse d'animation qu'il nomme «multiplan» («multiplane technique»).

En 1936, les studios Disney commencent à travailler sur leur premier long métrage: «Blanche Neige». Le travail de perspective sur les décors réussit à donner une impression de profondeur à un observateur fixe. Malheureusement, lorsqu'un zoom est réalisé sur l'image, l'accroissement de taille est le même pour les décors et pour les objets de premier plan. La notion de profondeur disparaît alors. Afin de remédier à ce problème, une machine nommée «multiplane camera» est construite. Les différents niveaux de l'image ne sont alors plus quasi-confondus dans le même plan mais se trouvent à des distances différentes de la caméra. De plus, afin de simuler un zoom, les différents plans ne se déplacent pas avec la même vitesse vers la caméra mais chacun a une vitesse inversement proportionnelle à la distance à l'observateur. Un décor supposé lointain s'approchera en conséquence moins vite que les objets de premier plan. On

peut constater dans le dessin-animé que cette simulation du déplacement des objets donne des résultats visuellement cohérents.

Dans [LEVO77], *Marc Levoy* reprend en conséquence cet astucieux principe de manière directe. La construction du décor peut alors être réalisé de deux manières différentes. Soit l'artiste crée directement un décor à deux dimensions, soit il crée une scène à trois dimensions dont le logiciel calculera une image afin de créer le décor. L'animation peut ensuite être recréée par le déplacement des différents plans d'image.

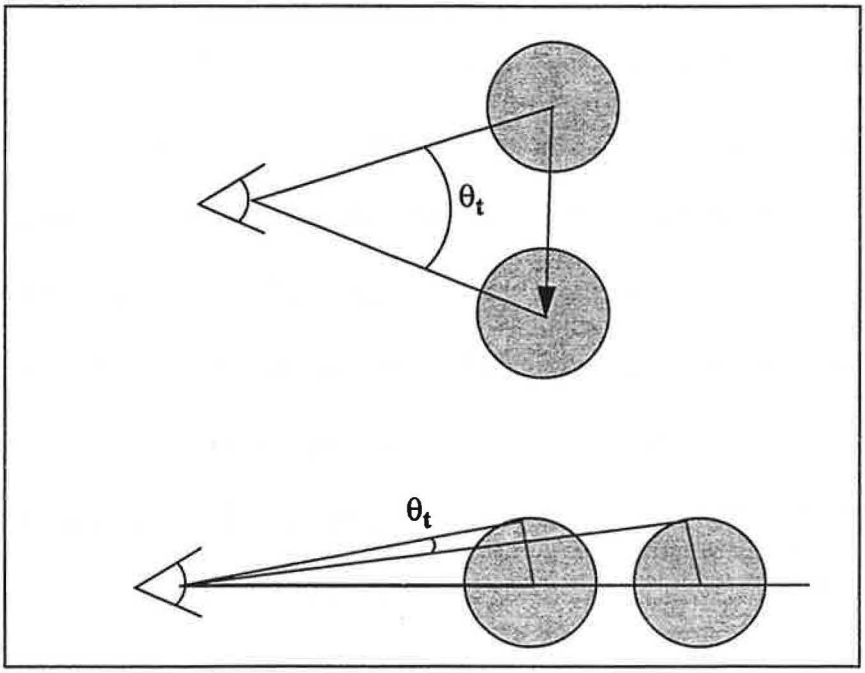
Il s'agit en conséquence ici d'une réelle imitation des processus de création manuelle d'animation, ce qui entraîne de nombreuses limitations. Les différents plans d'animation ne peuvent en effet interagir entre eux et doivent rester distincts: seules les parties cachées entre plans sont gérées. Un personnage ne peut en conséquence pas évoluer au milieu du décor. Il ne peut que se trouver devant ou derrière, à moins bien entendu de gérer les parties cachées manuellement. De même, la classification des objets en objets du décor ou en personnages est statique et laissée à l'interprétation de l'artiste.

Regan et Pose [REGA94] proposent en 1994 une méthode similaire mais avec l'optique de la synthèse d'images au sens actuel: c'est à dire qu'une fois la scène et les mouvements définis, le logiciel doit être capable de produire une animation de manière automatique. L'idée de départ est de classer les différents éléments de la scène ou objets en plusieurs classes selon la cohérence temporelle de leur apparence. Dans une animation réaliste, les objets ont en effet un mouvement continu. Ainsi, selon leurs mouvements et leur position, les objets peuvent avoir une apparence constante pendant une

certaine période. Il s'agit alors de détecter cette cohérence et de l'utiliser en laissant toute liberté de mouvement aux divers objets.

La segmentation des objets en différentes classes est réalisée dans l'espace des objets selon deux variations d'angle, celle due à un déplacement à distance constante de l'observateur et celle entraînée par la variation d'éloignement de l'objet. Afin de simplifier ce calcul, chaque objet est représenté par une sphère englobante.

Figure 14. Les critères de segmentation



En fixant un angle seuil, il est possible d'associer à chaque objet sa durée de validité, c'est à dire celle pendant laquelle le déplacement maximal ne dépasse pas l'angle de seuil. La segmentation est ensuite réalisée en fonction de cette durée de validité.

Chaque classe d'objets est traitée par le logiciel, qui fournit un résultat constitué d'une image de celle-ci ainsi que la carte de profondeur correspondante. L'image finale est enfin produite en mixant à chaque instant les différents résultats par la méthode classique du tampon de profondeur.

En proposant ainsi ce que l'on peut appeler des décors à trois dimensions, Regan et Pose résolvent le problème de l'indépendance des mouvements des objets entre les différentes classes et permettent un processus entièrement automatique.

Cependant, leur critère de segmentation ne prend en compte que les mouvements dans l'espace objet et ne permet pas de réellement conclure sur la modification ou non de l'apparence de l'objet une fois celui-ci projeté dans l'espace de l'image.

De plus, ce critère ne prend pas en compte les rotations possibles de l'objet autour du centre de la sphère englobante, ces rotations pouvant être le produit d'une rotation de l'objet ou encore du déplacement de l'observateur autour de celui-ci. Dans leur article, les auteurs affirment que ces mouvements sont négligeables. En effet, ils prennent comme exemple le déplacement d'un observateur dans une forêt. Les arbres possédant en général une bonne symétrie axiale autour de leur tronc, ce mouvement peut en conséquence passer inaperçu. C'est une remarque d'ailleurs utilisée pour les simulations temps réel, où les arbres sont souvent remplacés par une unique facette faisant face à l'observateur. Elle semble difficile à généraliser à tous les objets.

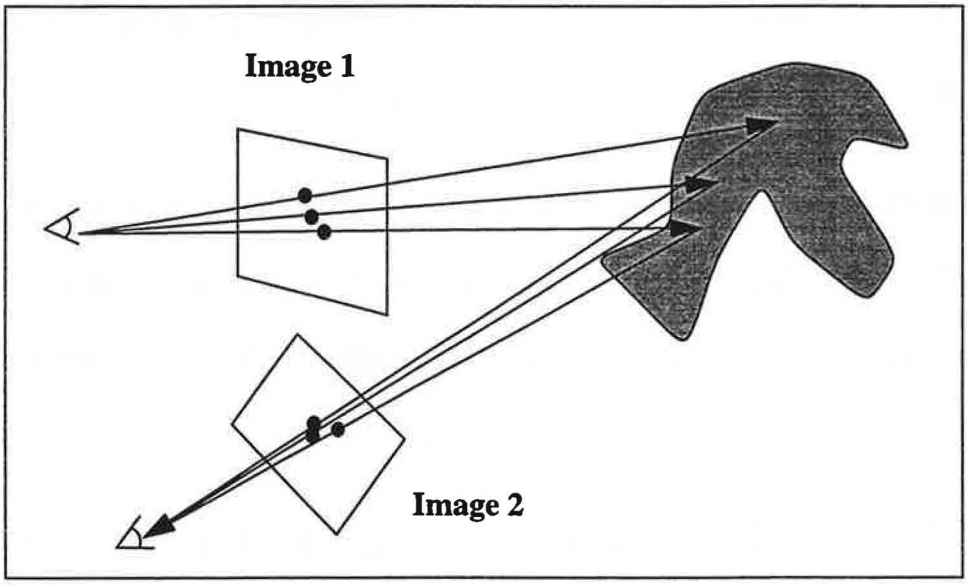
Enfin, la méthode de mixage des différents décors engendre de fortes discontinuités lors du remplacement d'un décor qui n'est plus valide par son remplaçant. Le complexe visuel est en effet très sensible à ce type de modifications brusques dans une image comme le fait remarquer Cook dans [COOK84].

I-3-3-2) Les méthodes de reprojection [BADT88] [ADEL95]

Si les méthodes à décor peuvent être considérées comme les héritières du dessin d'animation, les méthode de reprojection peuvent l'être de la stéréoscopie. Elles traitent en effet le problème de la visualisation d'une même scène sous différents angles.

Dans [BADT88], *Badt* propose de calculer une première image par une méthode classique puis d'en déduire l'image de la même scène après un déplacement de l'observateur. Lors du calcul préliminaire et pour chaque rayon primaire, on stocke à la fois la couleur du pixel résultant et les coordonnées à trois dimensions qui lui sont associées. Dans un second temps, chaque point à trois dimensions est reprojeté dans le plan de projection de la deuxième image.

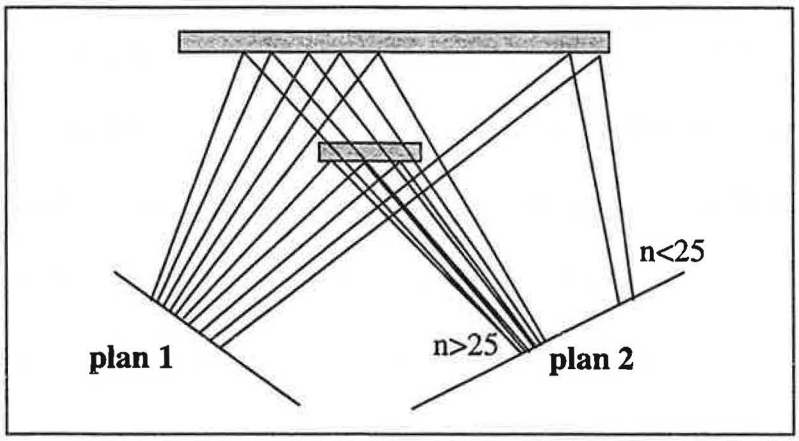
Figure 15. **Le principe de la reprojection**



La couleur d'un pixel final est obtenue en faisant la moyenne des couleurs des points qui s'y reprojettent. Pour chaque pixel, Badt propose de compter le nombre de points qui se reprojettent dans la région carrée définie par les 24 pixels voisins. Un nombre inférieur à 25 indique une zone cachée de l'image initiale qui s'est découverte

dans l'image finale. Au contraire, un nombre supérieur à 25 indique une zone masquée dans l'image finale et visible dans l'image initiale. Badt suggère de calculer les pixels des zones suspectes par un algorithme classique.

Figure 16. Les zones suspectes



Certains pixels peuvent cependant avoir un nombre proche de 25 sans qu'un point ne se reprojette en leur intérieur. *Badt* propose alors soit de lancer un nouveau rayon soit d'affecter au pixel une moyenne de la couleur de ses voisins.

Cette algorithme comporte de nombreuses limitations. Seul le mouvement de l'observateur est autorisé et la gestion des parties cachées est peu fiable. De plus, la couleur de la surface dépend de la position de l'observateur, ce qui n'est pas pris en compte ici.

Ainsi, *Adelson et al.* [ADEL95] proposent de modifier l'algorithme pour pallier ses faiblesses. Lors du calcul préliminaire, ils ne sauvegardent pas la couleur du pixel mais les informations qui permettent de la calculer: le point d'intersection 3D, la normale, la couleur diffuse et l'ombrage si les sources lumineuses sont statiques. La phase de reprojection est complétée par le déplacement du point d'intersection calculé en fonction des mouvements ou déformations de l'objet qui le porte. On vérifie ensuite que

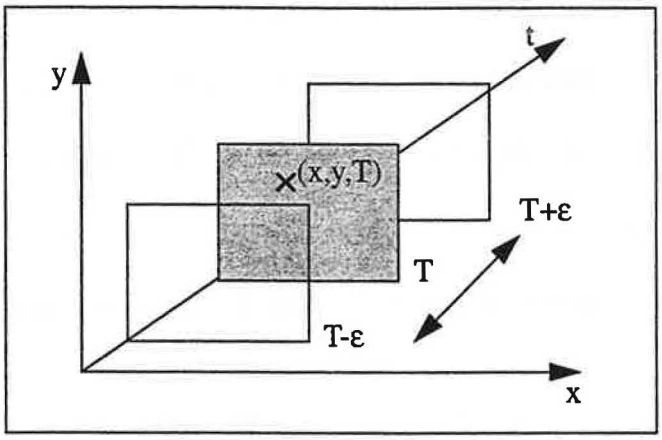
le point est toujours visible, afin de tenir compte des masquages dus aux mouvements de l'observateur et à celui des objets. De même, le rayon d'ombre doit être relancé si il y a eu déplacement du point d'intersection ou de la source lumineuse. Enfin, les phénomènes de réflection, réfraction, qui dépendent de la position de l'observateur, peuvent être pris en compte. De même que précédemment, il existe des points dans l'image finale qui ne sont pas fournis par la reprojection. Un calcul classique est alors réalisé.

Grâce à cette méthode, des images peuvent être produites avec des gains de temps que les auteurs évaluent entre 50 et 90% selon le type et le mouvement des objets. De plus, la reprojection n'implique pas que la couleur du pixel soit celle du rayon passant par son centre. L'aliassage est en conséquence réduit au profit du bruit comme le préconisait *Cook* avec son *lancer de rayons stochastique* [COOK86]. Cependant, cet algorithme travaille sur l'ensemble de la scène à chaque instant, ce qui peut dans certains cas être très lourd, notamment lors de la phase de vérification ou de lancer des rayons d'ombrage. De même, le déplacement du point d'intersection en fonction des déformations de l'objet peut être complexe.

I-3-3-3) L'échantillonnage spatio-temporel [NIME96]

Dans [NIME96], Nimeroff considère l'animation comme une succession temporelle d'images. Il crée un ainsi un espace à trois dimensions composé de la position du point dans les images et de sa coordonnée temporelle.

Figure 17. **L'espace spatio-temporel de l'animation**



Le calcul de l'animation est alors réalisé en deux étapes. En supposant l'animation connue a priori, il est possible d'échantillonner l'espace à trois dimensions. On tire au hasard un certain nombre de points dont on calcule les valeurs par une méthode de lancer de rayons classique. Afin d'éviter l'aliassage, un filtrage peut être réalisé: l'échantillon (x,y,t) sera alors une moyenne des rayons de ([x-dx,x+dx],[y-dy,y+dy],[z-dt,z+dt]).

La seconde phase est celle de reconstruction de toutes les images de l'animation. Supposons que l'on veuille calculer l'image à l'instant T. Un filtrage des échantillons est alors réalisé pour ne conserver que ceux dont la coordonnée temporelle appartient au segment [T-ε,T+ε] (Voir Figure ci-dessus). Les échantillons restants sont alors projetés dans l'image à calculer à x et y constants. On obtient ainsi un nuage de points sur lesquels il est possible de construire une triangulation de Delaunay. Une interpolation est ensuite réalisée à l'intérieur de chaque triangle.

Cette méthode permet de réduire le nombre de rayons effectivement lancés pour le calcul de l'animation. Elle n'est cependant pas sans défaut. La qualité de l'animation

n'est pas garantie car la détection des discontinuités ne l'est pas. Le nombre de points de l'échantillonnage est difficile à évaluer. De plus, l'animation doit être connue a priori.

I-3-3-4) Les méthodes de variation de couleur [BADT88] [CHAP90]

Les méthodes de variation de couleur sont fondées uniquement sur l'information de couleur des pixels de l'image.

Dans l'article [BADT88] qui a déjà été discuté ci-dessus, *Badt* proposait un second algorithme de ce type. Chaque pixel possède trois coordonnées qui sont sa ligne, sa colonne ainsi que le numéro de l'image dans laquelle il se trouve. La première image de l'animation est calculée par un lancer de rayons classique et est numérotée 0.

Supposons maintenant que les images 1 à (k-1) aient déjà été déduites par la méthode proposée et tentons de calculer l'image k. L'image k est initialement considérée comme identique à k-1. Des pixels sont ensuite choisis aléatoirement dans cette image et leur valeur réelle calculée. Si leur couleur réelle diffère de leur couleur supposée alors *Badt* considère qu'une certaine région autour du pixel est fausse. On remarquera qu'il ne précise pas, contrairement à [MAIL96] ce qu'il entend par la distance entre deux couleurs.

Les régions sont alors modifiées par une méthode dite «d'inondation», c'est à dire que tous les pixels voisins de ce pixel initial sont testés et ainsi de suite jusqu'à rencontrer des pixels considérés comme valides. Le voisinage des pixels comprend leurs trois dimensions, ce qui permet de revenir dans le passé pour détecter d'éventuels oublis antérieurs. On commence également à construire les images à partir de k+1. Les points

tirés aléatoirement dans les images suivantes sont alors choisis parmi les points non-encore inondés.

Afin de supprimer le stockage des images déjà calculées, *Badt* propose également une méthode de «demi-inondation» qui ne revient pas sur les images déjà existantes et ne réalise l'inondation que vers les images futures.

Cette méthode soulève de nombreuses interrogations notamment sur la distance entre couleurs choisie et sur la manière de tirer aléatoirement les points dans chaque image.

Chapman [CHAP90] propose une méthode similaire utilisant la dichotomie. Les pixels ont également trois dimensions, deux d'espace et une temporelle. Pour chaque position, on étudie la colonne de pixels indexée par le temps. Les couleurs d'instants pris dans la séquence sont calculées et comparées. Si leurs valeurs sont égales alors toute la colonne de pixels intermédiaires prend cette valeur. Sinon, le pixel médian est calculé et la méthode continue itérativement sur chacune des demi-colonnes.

Cette méthode, comme la précédente, se heurte au problème de la détection des petits objets. De plus, un objet même important qui traverse l'image laissera apparent le fond de celle-ci à ses deux instants extrêmes et ne sera en conséquence pas visualisé. Le choix de la hauteur initiale de la colonne est donc critique. Enfin pour être efficaces, ces algorithmes imposent un observateur fixe.

Pour toutes ces raisons, l'information qui peut être extraite uniquement des couleurs des images parait insuffisante pour la reconstruction d'une animation et ne peut s'appliquer qu'à des scènes très particulières.

I-3-4) La cohérence de l'arbre des rayons

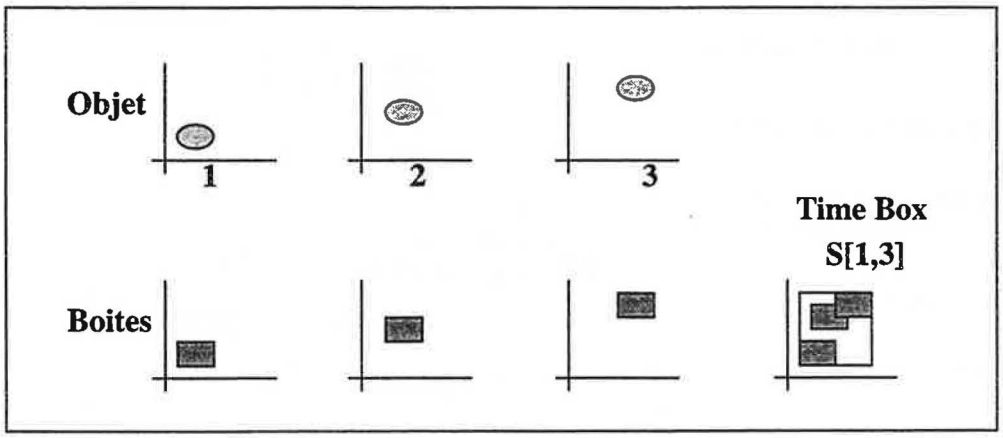
I-3-4-1) Une autre partition de l'espace temporel [GROL91]

Gröller et al. [GROL91] étendent la méthode de partition de l'espace de *Arvo et al.* [ARVO87] (Voir *Une partition de l'espace [ARVO87]*, page 20.) à la cohérence temporelle. Pour cela, ils ajoutent aux cinq dimensions des rayons décrites précédemment une sixième qui est le temps. Les sous-espaces sont donc de la forme:

$$Ei=[xi,x2]x[y1,y2]x[zi,z2]x[u1,v1]x[u2,v2]x[t1,t2]$$

Les objets gagnent eux aussi une dimension temporelle. Ceux-ci sont stockés sous forme d'un arbre CSG. Chaque feuille comporte la liste de ses transformations sous la forme d'une suite de matrices T1, T2,..., Tn. Afin d'accélérer le processus de détection des objets potentiellement intersectés, *Gröller* propose la notion de «time box», qui peut être vue comme une boîte englobante temporelle. Il s'agit en effet de construire un englobant des différentes positions d'un objet entre deux instants.

Figure 18. La construction d'une «time box»



Un arbre des «time box» est construit pour chaque feuille de l'arbre CSG. Cet arbre contient la «time box» de l'ensemble de l'animation ainsi que chacune des périodes obtenues en subdivisant récursivement la précédente par deux

Les «time box» de chaque primitive sont ensuite remontées tout au long de l'arbre CSG pour obtenir celles de chaque noeud de cet arbre. On ne calculera pas a priori toutes les «time box» de chacun de ces noeuds: si un arbre CSG représente un objet entre les instant i et j, on ne stockera que les boîtes [i,(i+j)/2] et [(i+j)/2+1,j]. On découpera ces boîtes seulement quand cela est nécessaire.

Figure 19. **L'arbre des «time box» associé à chaque feuille de l'arbre CSG**

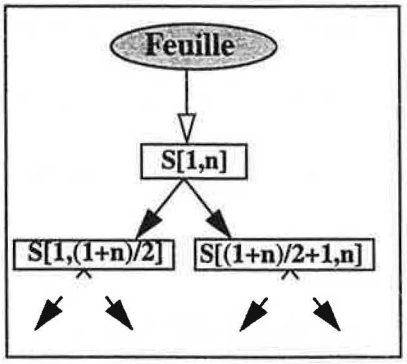
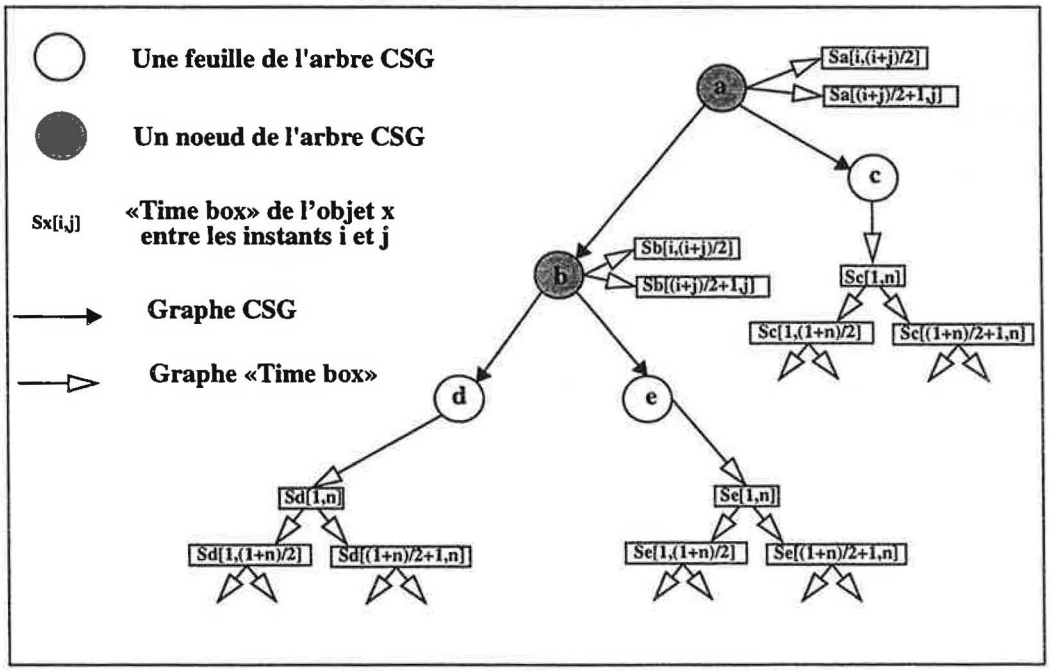


Figure 20. **L'extension temporelle de l'arbre CSG**



La méthode de partition de l'espace est ensuite exactement la même que celle décrite dans [ARVO87]. La classification des objets est alors faite entre le faisceau des rayons et les «time box» correspondantes.

L'accélération obtenue par cette méthode est de l'ordre de 2 dans les exemples choisis par les auteurs, alors qu' *Arvo et al.* [ARVO87] revendiquait une accélération de 10 par rapport à la méthode de l'octree [GLAS84]. On se heurte probablement ici à la complexité et à la taille des structures employées. Cette méthode demande de plus de connaître l'ensemble de l'animation à l'avance pour faire ce pré-traitement. On peut enfin lui faire le même reproche qu'à [GLAS88] qui recalcule l'apparence de tous les objets même si celle-ci est constante.

I-3-4-2) Les méthodes de conservation [SEQU89] [MURA90][MAUR93]

Les mises à jour incrémentales [SEQU89] [MURA90]

Le *lancer de rayons paramétré* [SEQU89] a pour objectif de rendre le travail de la personne qui modélise une scène le plus agréable possible. Très souvent en effet, le processus de modélisation est réalisé par itérations successives où l'on visualise à chaque fois une même scène où seuls les paramètres d'aspect (couleur, puissance des sources lumineuses, propriétés de la surface, etc...) sont modifiés. Il s'agit donc là de la version la plus élémentaire de la cohérence temporelle des rayons où aucun mouvement n'est autorisé.

Le principe de base est en conséquence de stocker l'arbre des rayons, où chaque noeud est une expression symbolique représentée par la liste de ses propriétés de surface. L'arbre est construit de manière classique lors du calcul de la première image puis on met à jour les paramètres nécessaires à chaque nouvelle modification. Le résultat de

chaque noeud peut également être stocké afin de ne pas recalculer une expression dont aucun coefficient n'a changé. Un drapeau indique alors s'il y a lieu ou non d'effectuer le calcul.

Le stockage de l'arbre demande une grande place mémoire. En conséquence Séquien et al proposent une méthode de hachage permettant d'utiliser la cohérence spatiale en partageant des sous-arbres identiques. La hachage des noeuds est fait par leurs propriétés uniquement géométriques. Ces propriétés (direction des sources, direction de l'observateur et vecteur normal) sont en effet les seules qui ne sont pas modifiées durant le processus de création de la scène. Les vecteurs sont normalisés. Tous les coefficients se trouvent alors dans [0,1], que l'on découpe en segments de taille ε. Deux coefficients situés dans le même segment sont considérés comme égaux. La fonction de comparaison entre noeuds s'étend, quant à elle, aux autre coefficients des propriétés de surface.

Cette technique est en conséquence rapide, mais ne considère que les modifications des paramètres d'aspect. Cette limitation est levée dans [MURA90] qui gère les déplacements des objets, toujours pour un observateur fixe, sous le nom de *lancer de rayons incrémental.*

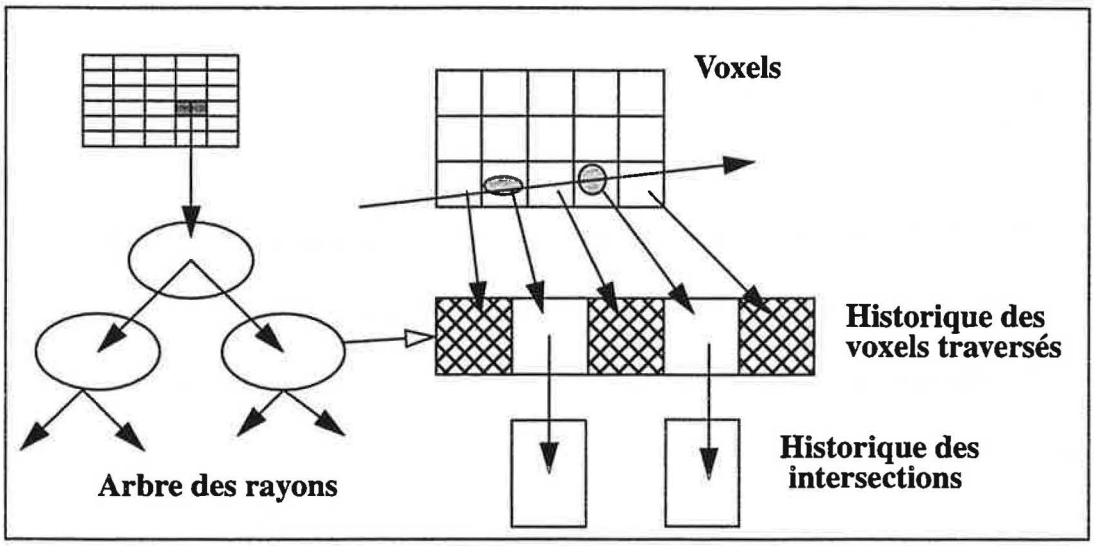
La philosophie de départ est la même que la précédente: l'arbre des rayons est créé au cours d'un premier calcul puis mis à jour pour chaque image. La particularité de la méthode est la voxelisation de l'espace. Chaque noeud de l'arbre contient en conséquence les informations suivantes:

- les paramètres du rayon (origine, destination).
- les informations sur le point d'intersection.
- l'historique des voxels traversés.

•une table de hachage.

L'historique des pixels traversés contient bien sûr la liste des voxels traversés. De plus, pour chaque voxel où le rayon rencontre un objet, un historique des intersections est disponible.

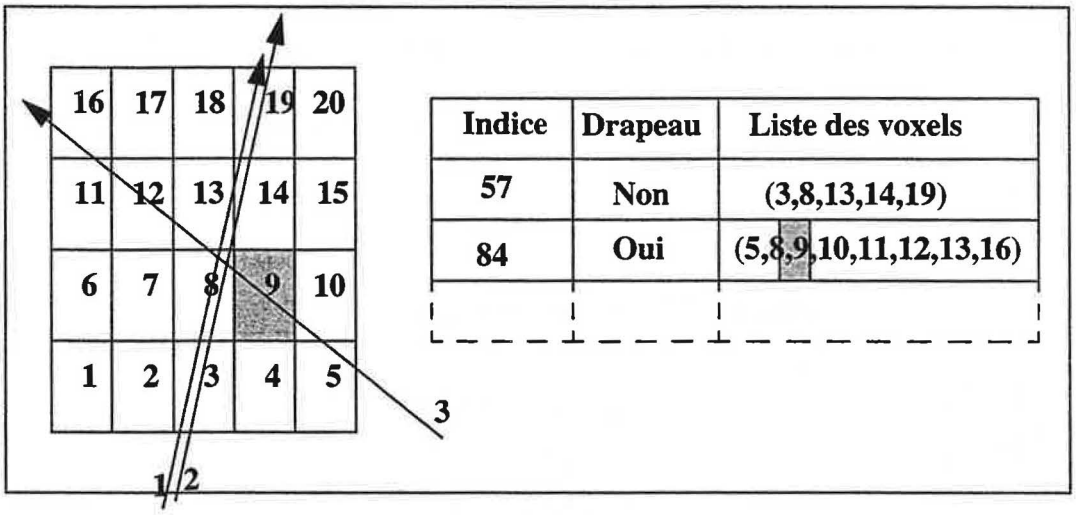
Figure 21. **L'historique des voxels**



L'historique des intersections contient la liste des objets intersectés. Pour chaque objet, on trouve les profondeurs associées à chaque intersection, c'est à dire la distance depuis l'origine du rayon. Si un objet change, seules ses intersections sont alors à modifier.

La table de hachage a pour but de détecter très rapidement quels sont les rayons qu'il est nécessaire de recalculer. Elle se présente comme suit:

Figure 22. **La table de hachage**



La fonction de hachage préconisée par les auteurs est la somme des indices des voxels traversés, la comparaison se faisant bien sur la liste complète ordonnée. Grâce à la cohérence des rayons, on limite le nombre d'indices dans la table: les rayons 1 et 2 ont par exemple le même indice.

Lors de la modification des objets, on détecte quels sont les voxels modifiés et on réalise le pré-traitement de la table de hachage: les drapeaux correspondant à une liste ou au moins un voxel est modifié sont marqués «Oui». Un rayon qui correspondra à cet indice de la table devra être recalculé. On pourra faire remonter ces drapeaux dans l'arbre des rayons afin d'accélérer son parcours: une branche dont aucune sous-branche n'est modifiée n'est en effet pas à parcourir.

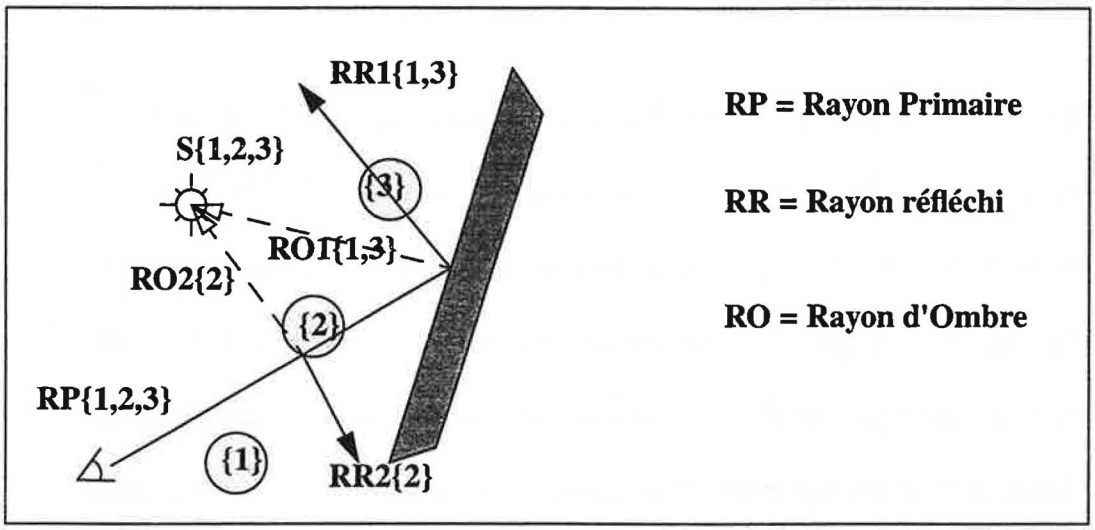
Cette méthode obtient de bons résultats en localisant les modifications de la scène. Elle nécessite cependant une programmation assez complexe et les structures sont coûteuses en mémoire. De plus, un rayon peut traverser un voxel modifié et donc nécessiter un nouveau calcul, sans que ses intersections soient réellement modifiées.

La factorisation des rayons [MAUR93]

Maurel et al. [MAUR93] traitent le problème de la conservation de l'arbre des rayons en factorisant les rayons ayant des propriétés identiques. Pour cela, ils donnent une dimension temporelle aux objets et aux rayons pour obtenir le *lancer de rayons 4D*.

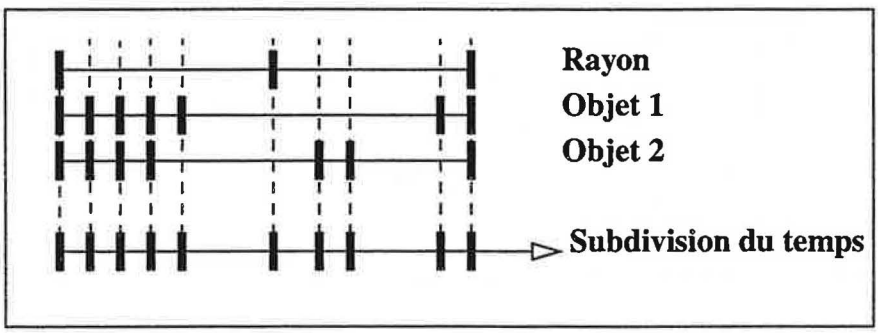
Chaque objet ou rayon dans cet espace est en conséquence décrit par sa géométrie accompagnée de la liste de ses instants de validité. Ces instant ne sont pas forcément successifs et seront notés entre accolades{}.

Figure 23. Un exemple du lancer de rayon 4D



Chaque objet a une liste d'instants de validité. On subdivisera en conséquence le temps en réalisant l'intersection des domaines de validité.

Figure 24. La subdivision du temps



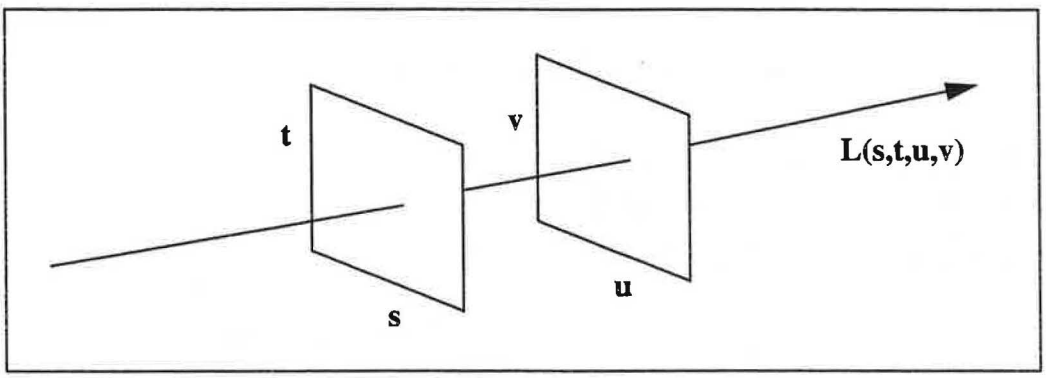
Dans chaque intervalle de la subdivision du temps, on cherchera enfin la description des rayons et des objets correspondante.

Il y a donc factorisation des calculs non seulement à l'intérieur du même intervalle mais également parce que la description d'un objet ou d'un rayon peut être identique sur des intervalles disjoints. Dans les exemple proposés, le calcul de 66% des rayons est évité grâce à cette méthode, ce qui est important. Cette méthode est cependant limitée à un observateur fixe et nécessite de plus la connaissance a priori de l'animation.

I-3-4-3) Les champs de lumière[LEVO96][GORT96]

Les champs de lumière sont à la cohérence des rayons ce que les décors étaient à la cohérence des images. Ils sont proposés simultanément dans [LEVO96] et sous le nom de lumigraph dans [GORT96]. Ce sont des structures accélératrices qui sont placées dans l'espace 3D. Il s'agit ici de pré-calculer les rayons traversant un volume de l'espace. Les rayons sont paramètrés par leurs intersections avec deux portions de plans parallèles. Dans [LEVO96], les portions de plans sont quelconques alors que dans [GORT96], elles sont présentées comme les faces d'un cube.

Figure 25. La paramètrisation des rayons

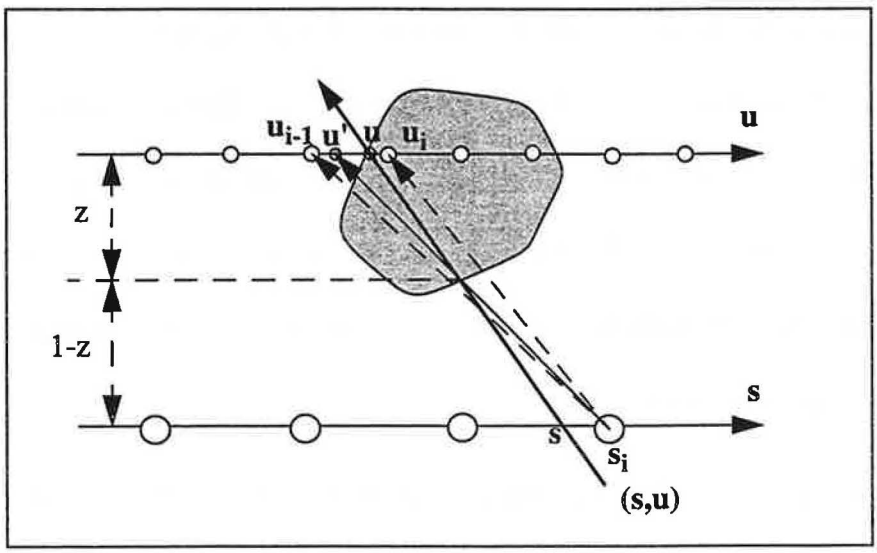


Le but est de pouvoir donner immédiatement sa valeur à chaque rayon entrant par la face (s,t). Comme il est impossible de pré-calculer toutes les valeurs de L(s,t,u,v), une discrétisation de chacun des plans est réalisée. Pour tout point de discrétisation de (s,t), un rayon est alors lancé vers chaque point de discrétisation de (u,v). Ce lancer est réalisé soit par un lancer de rayon classique, soit par des mesures physiques réalisées par une caméra.

Lorsqu'un rayon entre dans un champ de lumière, on cherche à lui affecter sa valeur. Supposons que le rayon ait pour coordonnées (s,t,u,v). La solution la plus simple est de lui affecter la valeur du rayon (s_i, t_j, u_p, v_q), où (s_i, t_j) {resp. (u_p, v_q)} est le point de discrétisation le plus proche de (s,t) {resp. (u,v)}. Un filtrage est également possible, soit bi-linéaire sur les valeurs (s,t) ou (u,v), soit quadri-linéaire sur les deux.

En cas d'intersection avec un objet présent dans le champ de lumière, le rayon le plus proche à l'entrée et à la sortie du champ ne donne cependant pas obligatoirement l'intersection la plus proche de la valeur réelle. Dans la figure ci-dessous par exemple, le rayon s_i, u_{i-1} semble une solution meilleure que la solution directe s_i,u_i.

Figure 26. La correction de l'intersection



Gortler et al. proposent en conséquence une correction de la valeur de u. En facétisant l'objet, il est rapide de calculer la valeur z de l'intersection. Pour chaque s_i, on peut calculer alors la valeur u' d'un rayon passant par s_i et l'intersection.

$$u' = u + (s - s_i) \times \frac{z}{1 - z}$$

Le rayon le plus proche devient alors (s_i, u_{i-1}), ce qui est ce que nous voulions. L'opération sera répétée sur les axes (t,v) et autant de fois que nécessaire, soit par exemple huit fois (2x4) pour une interpolation quadri-linéaire.

Cette méthode a l'avantage de pouvoir mêler des objets réels et virtuels pour produire des animations de manière rapide. Elle nécessite cependant un pré-calcul ou une saisie physique relativement lourds. Il faut environ une heure pour saisir un objet physique. Enfin, le champ de lumière ne reste valide que si l'illumination qu'il reçoit n'est pas modifiée.

I-4) Conclusion

De nombreuses techniques ont été proposées en vue d'accélérer le calcul des animations de synthèse. Un résumé des propriétés et limitations de celles que nous avons exposées précédemment est présenté ci-dessous sous la forme d'un tableau. Nous avons décomposé [BADT88] en deux: [BADT88]1 est l'algorithme de reprojection, [BADT88]2 celui utilisant uniquement les couleurs des pixels. Une croix indique si l'algorithme répond aux différents critères:

- Les mouvements des objets, de l'observateur et des sources sont-ils possibles?

•La qualité de l'animation est elle conservée?

•Peut-on utiliser cette technique pour le temps réel où il n'existe pas de connaissance a priori de l'animation?

TABLEAU 1. Résumé des techniques d'accélération du calcul d'animation

	Mouvement objets	Mouvement observateur	Mouvement source	Qualité	Connaissance a priori inutile
[GLASS88]	X	X	X	X	
[BADT88]1	X	X			X
[ADEL95]	X	X	X	X	X
[BADT88]2	X				X
[NIME96]	X	X	X		
[CHAP90]	X				
[LEVO77]	limité				
[REGA94]	X	X			X
[GROL91]	X	X	X	X	
[SEQU89]				X	
[MURA90]	X		X	X	X
[MAUR93]	X		X	X	
[LEVO96] [GORT96]	X	X		X	X après pré-calcul

[ADEL95] apparaît ainsi comme une bonne solution au problème, mais le fait de travailler sur l'ensemble de la scène limite son efficacité réelle.

Notre objectif dans la suite de ce manuscrit sera en conséquence de proposer un algorithme d'accélération qui réponde aux critères du tableau ci-dessus. Nous choisirons pour cela une méthode qui permette d'évaluer l'impact des mouvements des objets sur les images produites, On tentera ainsi de ne traiter à chaque image que les objets pertinents et non l'ensemble de la scène.

CHAPITRE II

De la mesure également

II-1) Introduction

Nous avons décrit dans le chapitre précédent l'importance de la notion de cohérence temporelle et les applications qui en ont été faites jusqu'à aujourd'hui, que ce soit dans le domaine de la compression de sources vidéo ou dans celui de la création d'animations classiques ou de synthèse. Dans le domaine qui nous intéresse, celui du rendu d'animations de synthèse, une question importante se dégage: celle de la détection des modifications entre images successives de l'animation.

Souvent ([REGA94] [GLAS88] [GROL91]), on ne tient pas compte de l'importance relative de chaque objet dans l'image finale. Un objet occupant un grand nombre de pixels demandera en effet le même travail vis à vis de la construction de la structure accélératrice qu'un objet occupant moins d'un pixel. Dans [BADT88] et [CHAP90] au contraire, on ne se préoccupe que de la couleur locale du pixel, sans aucune notion de topologie.

Les algorithmes qui réalisent une mise à jour incrémentale de l'arbre des rayons [MURA90] [MAUR93] imposent un observateur fixe, ce qui réduit de manière importante leur domaine d'application.

De plus, dans certains algorithmes, la connaissance a priori de l'animation est nécessaire, soit pour construire une structure accélératrice préalable ([GLAS88], [GROL91][MAUR93]), soit pour déterminer quelles sont les zones de l'image qu'il est nécessaire de recalculer ([BADT88], [CHAP90]). Dans de nombreux cas cependant, la connaissance a priori de l'animation n'est pas disponible. C'est le cas notamment dans les nombreuses applications de la simulation temps-réel comme les simulateurs ou les jeux vidéos.

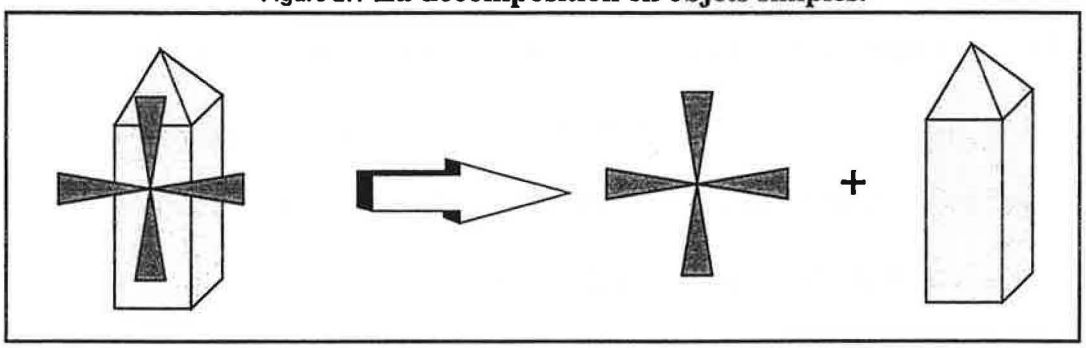
Pour toutes ces raisons, nous décrirons dans ce chapitre une méthode qui a pour but de mesurer les modifications de l'apparence de l'objet induites par un déplacement dans l'espace objet. On autorisera les mouvements de l'observateur et on s'affranchira de la connaissance a-priori de l'animation par une prédiction du mouvement des objets.

II-2) Définitions préalables et conventions

Dans la suite de ce manuscrit, le terme *objet* désignera une union de primitives (facettes ou primitives CSG) indéformables et ne possédant aucun degré de liberté entre elles. La construction de la liste des objets appartient à la phase de modélisation où cette notion d'objet apparaît de manière naturelle. Comme dans le dessin en deux dimensions, il y est courant de grouper des primitives entre elles afin de facilement les positionner ou de les copier à de multiples exemplaires. Il apparaît en conséquence aisé de

modifier un modeleur afin qu'il fournisse une liste de ces objets. Ainsi, un moulin à vent sera composé de deux objets: la tour et les ailes.

Figure 27. **La décomposition en objets simples.**



Comme il parait utopique de tenir compte de la géométrie complète de chaque objet dans la détection de ses mouvements, une sphère englobante, décrite par son centre et son rayon, sera associée à chaque objet.

Dans la suite de ce chapitre, on supposera que l'observateur se trouve à l'extérieur de la sphère englobante de l'objet. En effet, les cas où l'observateur se trouve à l'intérieur sont rares et aucune conclusion sur le mouvement ne peut alors être donnée, à moins que l'objet soit immobile! Ce cas sera néanmoins traité dans le chapitre suivant.

Puisque nous nous intéressons aux modifications apparentes de chaque objet dans l'image produite, tous les mouvements décrits dans la suite de ce chapitre seront des mouvements relatifs: il convient d'ajouter aux mouvements propres de l'objet dans son repère local ceux de l'observateur.

II-3) L'exposé belge

Nous nous intéressons ici au cas de la projection perspective. La projection d'une sphère sur l'écran est alors l'intersection entre le cône de sommet l'observateur, tangent à la sphère, et le plan de projection. Par définition, cette intersection est une conique. Nous allons tout d'abord exposer une méthode permettant de définir cette conique dans l'espace puis nous décrirons les calculs qui en découlent.

Le lien entre la définition bifocale d'une conique et la définition sous forme de section de cône de révolution a été fait par les belges Dandelin et Quételet dans un exposé auquel Michel Berger donne le nom d'exposé belge [BERG78].

Cet exposé nous permet de définir les foyers de l'ellipse comme le point de tangence entre deux sphères et le plan de section comme dans les figures suivantes:

Figure 28. L'ellipse comme section d'un cône de révolution

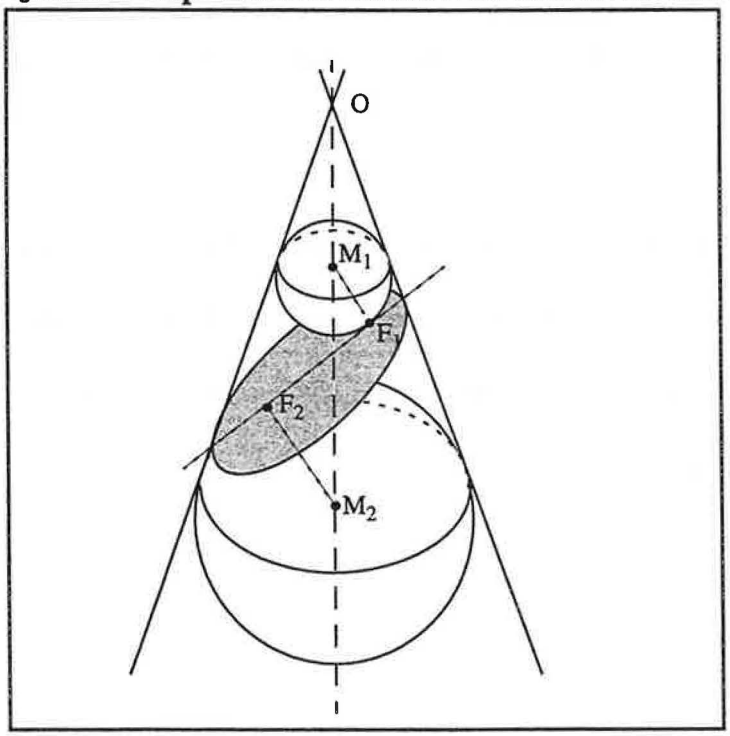
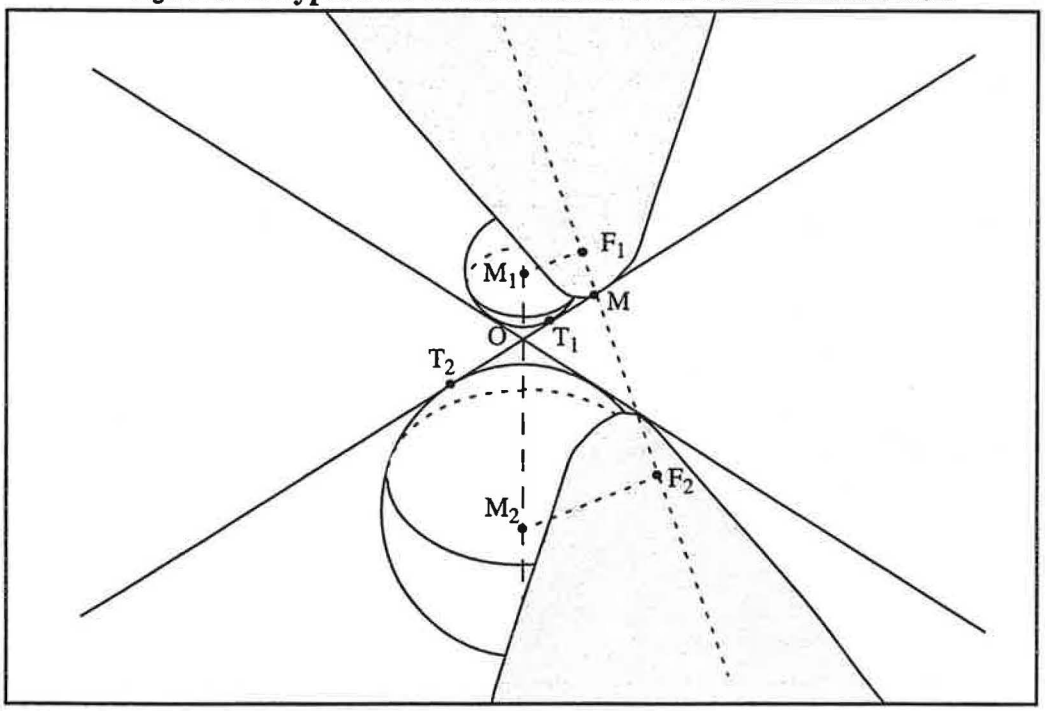


Figure 29. **L'hyperbole comme section d'un cône de révolution**

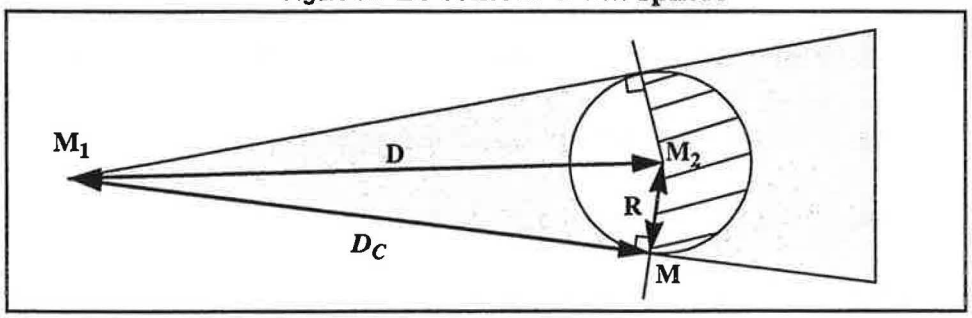


Le cas de la parabole est bien entendu également possible comme cas limite entre l'ellipse et l'hyperbole. Dans la suite de ce chapitre, ce cas sera traité comme celui de l'hyperbole.

Lemme 1:

Tous les segments de tangentes issus d'un même point à une sphère ont la même longueur.

Figure 30. **Le contour d'une sphère**



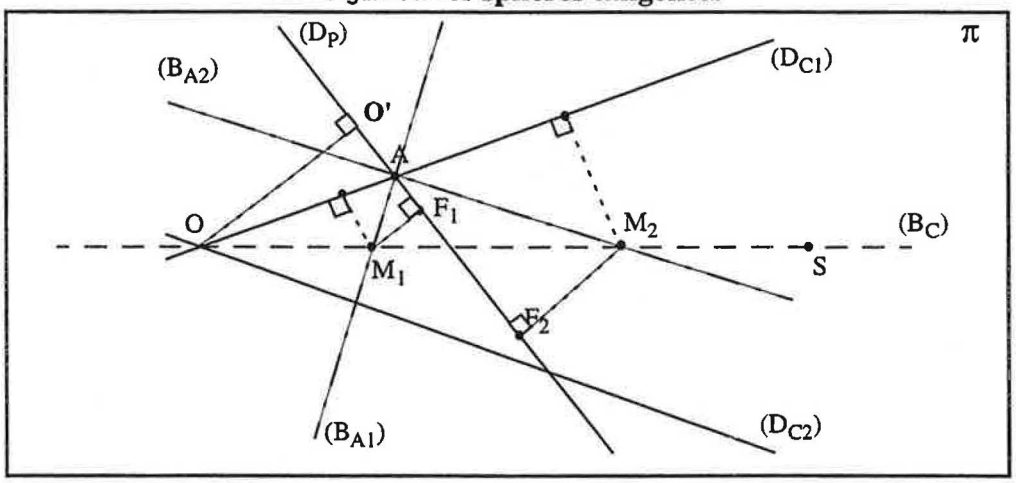
On déduit de la figure ci-dessus la distance entre M_1 et le contour de la sphère:

$$D_C^2 = D^2 - R^2 \qquad \text{(EQ 1)}$$

Lemme 2:

Il existe toujours au moins une sphère inscrite dans le cône et celle-ci est tangente au plan de section.

Figure 31. les sphères tangentes



Plaçons nous comme dans la figure 31, page 56 dans le plan π défini par le sommet du cône O, sa projection orthogonale sur le plan de section O' et le centre de la sphère S. Ce plan et le plan de section se coupent selon une droite (D_P). Le cône, symétrique par rapport à π, le coupe en deux droites non-parallèles (D_{C1}) et (D_{C2}) (sauf dans le cas d'un cône réduit à une droite, que nous ne traiterons pas!). (D_P) coupe au moins une des deux droites (D_{C1}) et (D_{C2}) en un point unique A, puisque celles-ci ne sont pas parallèles. Supposons que A soit l'intersection de (D_P) et de (D_{C1}).

Soit alors (B_C) la bissectrice de l'angle formé par les droites (D_{C1}) et (D_{C2}) comprise dans le cône, c'est à dire l'axe de symétrie du cône. Soient (B_{A1}) et (B_{A2}) les bissectrices des angles formés par (D_P) et (D_{C1}) en A. (B_C) intercepte au moins une de ces deux bissectrices puisque celles-ci sont perpendiculaires. Soit M le point d'intersection. Ce point, intersection des bissectrices, se trouve à égale distance de (D_P), (D_{C1}) et (D_{C2}). Au regard de la symétrie du cône, on peut conclure que ce point est à égale dis-

tance du cône et du plan de section: c'est le centre d'une sphère tangente au plan et inscrite dans le cône.

Remarque: dans le cas général, il existe deux sphères tangentes au plan et inscrites dans le cône. Le cas de la sphère unique correspond à la parabole.

Propriété:

Les points de tangence des sphères au plan de section sont les foyers de la conique intersection du cône et du plan.

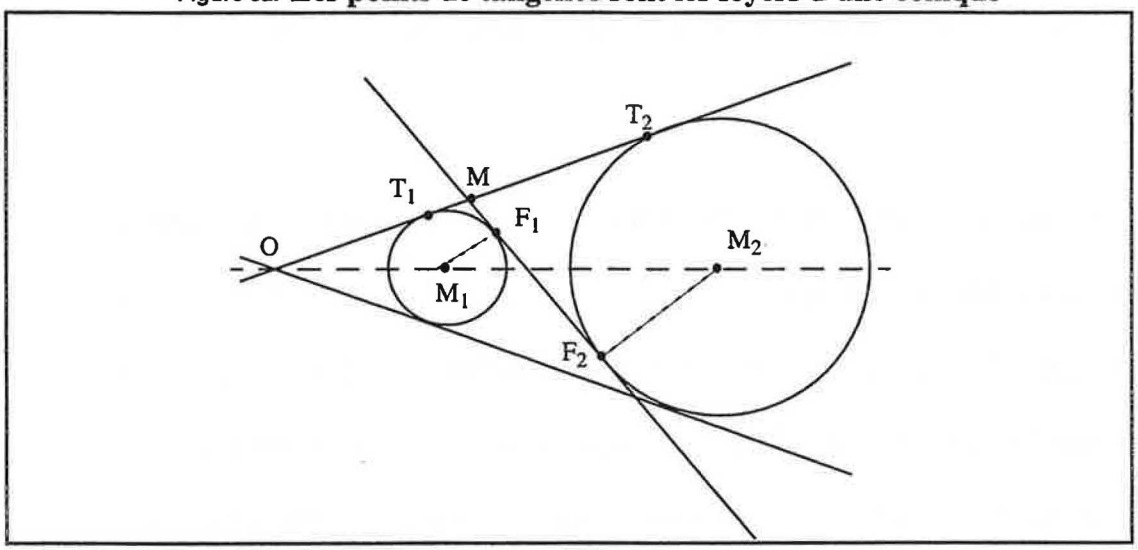
Démonstration:

Soient F_1 et F_2 les points de tangence des sphères au plan de section.

Soit M un point de l'intersection entre le cône et le plan de section.

On peut alors définir les points T_1 et T_2 comme les points de tangence aux sphères appartenant à la droite reliant M au sommet du cône O.

Figure 32. Les points de tangence sont les foyers d'une conique



Selon le lemme 1, on sait que: $MF_1 = MT_1$ et $MF_2 = MT_2$.

☛ Dans le cas où les deux sphères se trouvent dans le même demi-cône (figure 32, page 57), on a: $MT_1 = OM - OT_1$ et $MT_2 = OT_2 - OM$, d'où:

$$MF_1 + MF_2 = MT_1 + MT_2 = (OM - OT_1) + (OT_2 - OM) = OT_1 + OT_2 .$$

Or, toujours selon le lemme 1, OT_1 ne dépend pas du point T_1: c'est la distance de O au contour de la première sphère. De même, OT_2 est indépendant du choix du point M sur l'intersection.

On en déduit en conséquence que $MF_1 + MF_2$ est une constante, quel que-soit le choix du point M: l'intersection est une ellipse de foyers F_1 et F_2.

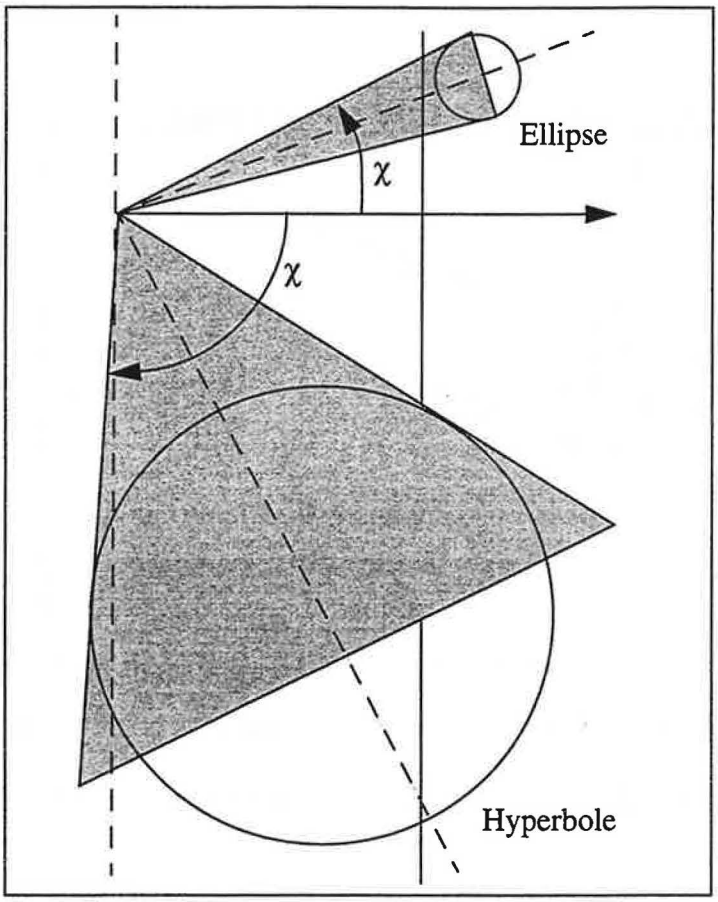
Remarque importante:

On déduit également des démonstrations précédentes que la projection orthogonale du sommet du cône sur le plan de section se trouve sur la droite définie par le grand axe de l'ellipse.

☛ Dans le cas de la Figure 29, "L'hyperbole comme section d'un cône de révolution", page 55, on pourrait démontrer de la même manière que $| MF_1 - MF_2 |$ est une constante.

Lorsque la conique projetée est une hyperbole, aucune information ne peut être fournie sur son mouvement puisque certains de ses points se trouvent à l'infini. Ce cas se produit lorsque l'angle χ de plus grande ouverture du cône est supérieur à $\pi/2$ (voir figure 33, page 59). Ce cas est peu fréquent puisque la plupart des objets qui nous intéressent se trouvent dans le cône de projection. Ainsi, dans la suite de ce chapitre, nous ne nous intéresserons qu'au cas où la projection de l'objet est une ellipse.

Figure 33. **Le type de la conique**



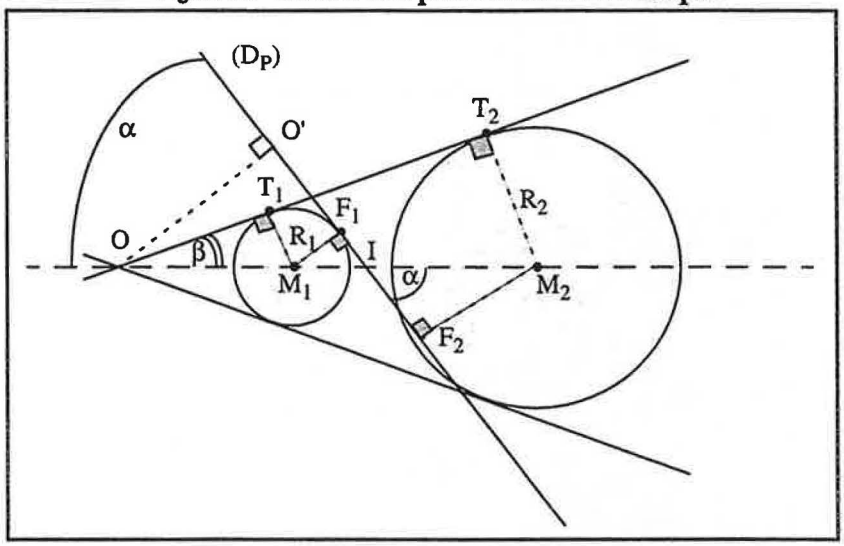
II-4) Comment mesurer les translations

II-4-1) Introduction

On cherchera dans cette partie à calculer les conséquences des déplacements d'un objet dans l'espace en terme de translations de sa projection dans l'espace image, afin de mesurer leur impact sur l'image. Puisque l'on ne s'intéresse qu'au cas où cette projection est une ellipse, on doit définir un moyen pour calculer les différentes positions de celle-ci. C'est ce qui est fait dans les paragraphes suivants.

II-4-2) Le calcul des paramètres de l'ellipse

Figure 34. **Calcul des paramètres de l'ellipse**



Nous reprenons ici les propriétés de l'ellipse comme intersection d'un cône et d'un plan du paragraphe précédent. Les notations seront celles de la figure 34, page 60.

II-4-2-1) Calcul des sphères tangentes:

Dans les triangles rectangles, on a:

$$\sin\alpha = \frac{R_1}{IM_1} = \frac{R_2}{IM_2} = \frac{OO'}{OI}$$

$$\sin\beta = \frac{R_1}{OM_1} = \frac{R_2}{OM_2} = \frac{R}{D}$$

où: R est le rayon de la sphère englobante définissant le cône

D est la distance entre l'observateur et le centre de la sphère englobante que l'on nommera S.

O' est la projection orthogonale de O sur le plan de section.

I est l'intersection entre le plan de section et la droite (OS).

Toutes ces valeurs sont connues dès que l'on se donne la sphère englobante, l'observateur et le plan de projection.

☛ Pour la petite sphère, on a $OI = OM_1 + M_1I$ soit

$$OI = R_1 \cdot \frac{D}{R} + R_1 \cdot \frac{OI}{OO'} = R_1 \cdot \left(\frac{D}{R} + \frac{OI}{OO'}\right) = R_1 \cdot \left(\frac{1}{\sin\beta} + \frac{1}{\sin\alpha}\right)$$

d'où l'on peut déduire R_1 puis OM_1 et enfin la position du point M_1 dans l'espace puisque:

$$\overrightarrow{OM_1} = OM_1 \cdot \frac{\overrightarrow{OS}}{D}$$

☛ Pour la grande sphère, on a $OI + IM_2 = OM_2$, d'où l'on déduit:

$$OI = R_2 \cdot \left(\frac{D}{R} - \frac{OI}{OO'}\right) = R_2 \cdot \left(\frac{1}{\sin\beta} - \frac{1}{\sin\alpha}\right)$$

d'où la description de la seconde sphère.

Remarque: le cas ($OI < 0$) correspond à ($\sin\beta > \sin\alpha$) soit ($\beta > \alpha$) qui est le cas de l'hyperbole.

Après simplification, on obtient:

$$\overrightarrow{OM_1} = \frac{OO'}{(\sin\alpha + \sin\beta)} \cdot \frac{\overrightarrow{OS}}{D}$$

$$\overrightarrow{OM_2} = \frac{OO'}{(\sin\alpha - \sin\beta)} \cdot \frac{\overrightarrow{OS}}{D}$$

avec

$$\sin\alpha = \frac{\overrightarrow{OS} \cdot \overrightarrow{OO'}}{D \times OO'}$$

$$\sin\beta = \frac{R}{D}$$

soit encore

$$\overrightarrow{OM_1} = \frac{OO'}{(A+B)} \cdot \overrightarrow{OS}$$

$$\overrightarrow{OM_2} = \frac{OO'}{(A-B)} \cdot \overrightarrow{OS}$$

avec

$$A = \frac{\overrightarrow{OS} \cdot \overrightarrow{OO'}}{OO'}$$

$$B = R$$

II-4-2-2) Calcul des points particuliers de l'ellipse

Les foyers de l'ellipse sont simples à calculer puisque ce sont les projections des centre des sphères tangentes sur le plan de section. Afin de décrire totalement l'ellipse, nous allons maintenant calculer les points extrêmes de son grand axe.

Ces points se trouvent sur la droite définie par les deux foyers: ce sont les intersection entre cette dernière et le cône tangent à la sphère.

☛ La définition du cône:

Un point M du cône vérifie que l'angle (OM,OS) est constant ce que l'on peut écrire:

$$\frac{|\overrightarrow{OM} \cdot \overrightarrow{OS}|}{OM \cdot OS} = K_2$$

soit pour la partie du cône qui se trouve du même côté que la sphère:

$$\overrightarrow{OM} \cdot \overrightarrow{OS} = K \cdot OM$$

Soit alors T un point de tangence entre le cône et la sphère:

$$\overrightarrow{OT} \cdot \overrightarrow{OS} = OT \cdot OS \cdot \cos\beta = OT \cdot OS \cdot \frac{OT}{OS} = OT^2$$

T appartient au cône, il vérifie en conséquence son équation:

$$\overrightarrow{OT} \cdot \overrightarrow{OS} = K \cdot OT = OT^2$$

d'où:

$$K = OT = \sqrt{OS^2 - R^2}$$

d'où l'équation du cône:

$$\boxed{\overrightarrow{OM} \cdot \overrightarrow{OS} = \sqrt{OS^2 - R^2} \cdot OM}$$

☛ Paramètrage de la droite support du grand axe

Soit E le milieu du segment reliant les foyers F_1 et F_2. E est le centre de symétrie de l'ellipse. Un point M de la droite support du grand axe vérifie:

$$\overrightarrow{EM} = \lambda \cdot \overrightarrow{F_1F_2}$$

☛ Le calcul des extrémités du grand axe

Soit M le point d'intersection du cône et du grand axe:

$$\overrightarrow{OM} \cdot \overrightarrow{OS} = K \cdot OM$$

Avec la paramètrisation de la droite, la partie gauche de l'équation devient:

$$\overrightarrow{OM} \cdot \overrightarrow{OS} = (\overrightarrow{OE} + \overrightarrow{EM}) \cdot \overrightarrow{OS} = \overrightarrow{OE} \cdot \overrightarrow{OS} + \lambda \cdot \overrightarrow{F_1F_2} \cdot \overrightarrow{OS}$$

et la partie droite de l'équation au carré:

$$K^2 \cdot OM^2 = K^2 \cdot (\overrightarrow{OE} + \overrightarrow{EM})^2 = K^2 \cdot (\overrightarrow{OE} + \lambda \cdot \overrightarrow{F_1F_2})^2 = K^2 \cdot (OE^2 + 2\lambda \cdot \overrightarrow{OE} \cdot \overrightarrow{F_1F_2} + \lambda^2 \cdot {F_1F_2}^2)$$

En élevant les deux membres de l'équation au carré, on obtient une équation du second degré dont les solutions sont opposées puisque E est le centre de symétrie de l'ellipse. Cette équation résultante ne comprend donc pas de terme de degré un.

$$(\overrightarrow{OE} \cdot \overrightarrow{OS})^2 + \lambda^2 \cdot (\overrightarrow{F_1F_2} \cdot \overrightarrow{OS})^2 = K^2 \cdot (OE^2 + \lambda^2 \cdot {F_1F_2}^2)$$

soit

$$\lambda^2 = \frac{K^2 \cdot OE^2 - (\overrightarrow{OE} \cdot \overrightarrow{OS})^2}{(\overrightarrow{F_1F_2} \cdot \overrightarrow{OS})^2 - (K^2 \cdot {F_1F_2}^2)}$$

ce qui permet de calculer les extrémités du grand axe.

☛ Le calcul des extrémités du petit axe

La droite support du petit axe est à la fois perpendiculaire à celle support du grand axe et au vecteur de visée de l'observateur puisqu'elle appartient au plan de projection.

On peut en conséquence définir un vecteur directeur de cette droite comme le produit vectoriel entre le vecteur défini par les foyers de l'ellipse et le vecteur de visée. On procède ensuite comme pour le grand axe.

$$\vec{V} = \overrightarrow{F_1F_2} \wedge \overrightarrow{OO'}$$

$$\overrightarrow{EM} = \lambda \cdot \vec{V}$$

$$\lambda^2 = \frac{K^2 \cdot OE^2 - (\overrightarrow{OE} \cdot \overrightarrow{OS})^2}{(\vec{V} \cdot \overrightarrow{OS})^2 - (K^2 \cdot V^2)}$$

Il faut remarquer ici que les numérateurs sont identiques pour les deux expressions de λ, ce qui réduit les calculs à effectuer.

II-4-3) L'évaluation des mouvements de translation

La connaissance de l'ellipse permet d'évaluer les mouvements de translation de l'objet dans l'espace image. Celle-ci peut, en effet, être définie par la connaissance des quatre sommets de ses axes. Puisque nous pouvons connaître les positions de ces points dans l'espace image au cours du temps, on définira le déplacement en translation de l'ellipse entre deux instants comme le maximum des déplacements de ces points.

II-5) Comment mesurer les rotations

II-5-1) Introduction

Un objet peut être soumis à des rotations pour deux raisons. La première est une rotation de l'objet dans son repère local. Dans un second cas, beaucoup plus fréquent, c'est l'observateur qui au cours de son mouvement, provoque une rotation apparente de l'objet. On peut prendre comme exemple un promeneur dans une forêt qui découvre différentes faces d'un arbre en passant à ses côtés. Cette rotation s'accompagne alors d'une translation dans son champ de vision, mais les deux ne sont pas forcément liées. Le même promeneur qui observe un oiseau perché sur une branche tournera la tête lors de son déplacement. La position apparente de l'oiseau peut ne pas être modifiée mais son apparence évoluera au cours du déplacement.

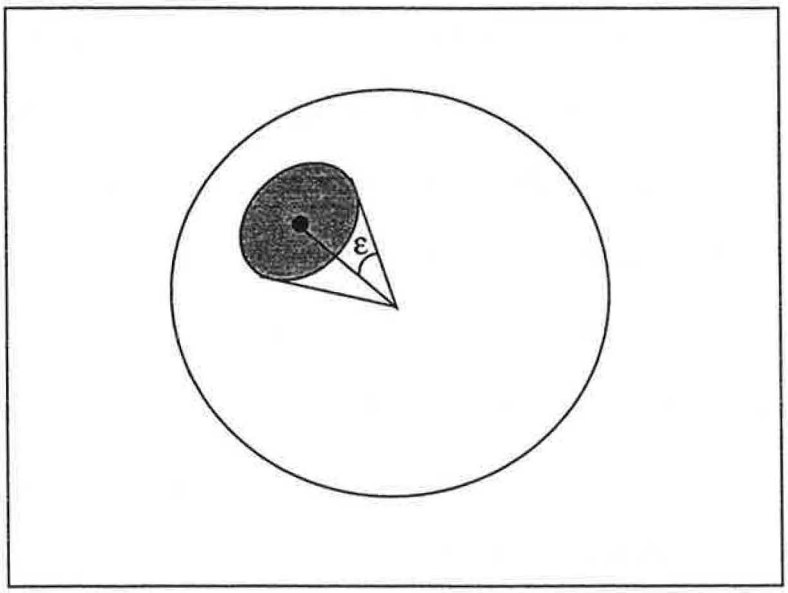
On cherchera en conséquence dans cette partie à calculer les conséquences des déplacements d'un objet dans l'espace en terme de rotation de sa projection dans l'espace image, afin de mesurer leur impact sur l'image. Pour ce faire, on cherchera tout d'abord quels sont les points sur la sphère qui maximalisent cet impact.

II-5-2) La recherche des points maximaux

II-5-2-1) Position du problème: la calotte sphérique

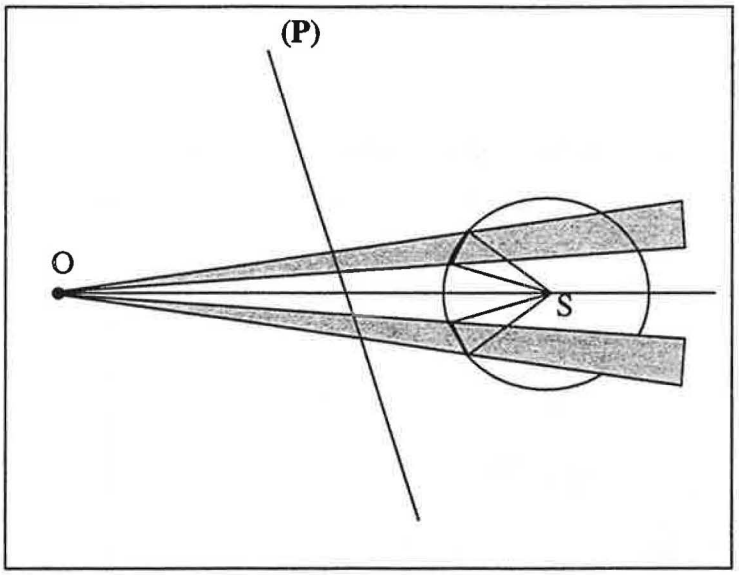
Considérons un point à la surface de la sphère. Une rotation de celle-ci fait décrire un arc de cercle à ce point. C'est cet arc une fois projeté qui mesurera le déplacement engendré dans l'espace image par la rotation de la sphère. Comme les rotations peuvent avoir lieu dans toutes les directions autour du point, on s'intéressera à toutes les nouvelles positions possibles de ce point après rotation d'un angle constant en valeur absolue, qui est une calotte sphérique autour du point.

Figure 35. La calotte sphérique de positions possibles.



Considérons maintenant la projection de cette calotte dans le plan de l'image

Figure 36. **La projection de la calotte sphérique**



La sphère étant bien entendu symétrique autour de (OS), toutes les calottes obtenues par rotation d'un point autour de cet axe sont identiques. Il vient par conséquence comme dans la figure 36, page 67 que les cônes de projection sont identiques. Dans la suite, l'ensemble de ces points obtenu par rotation sera dénommé «couronne».

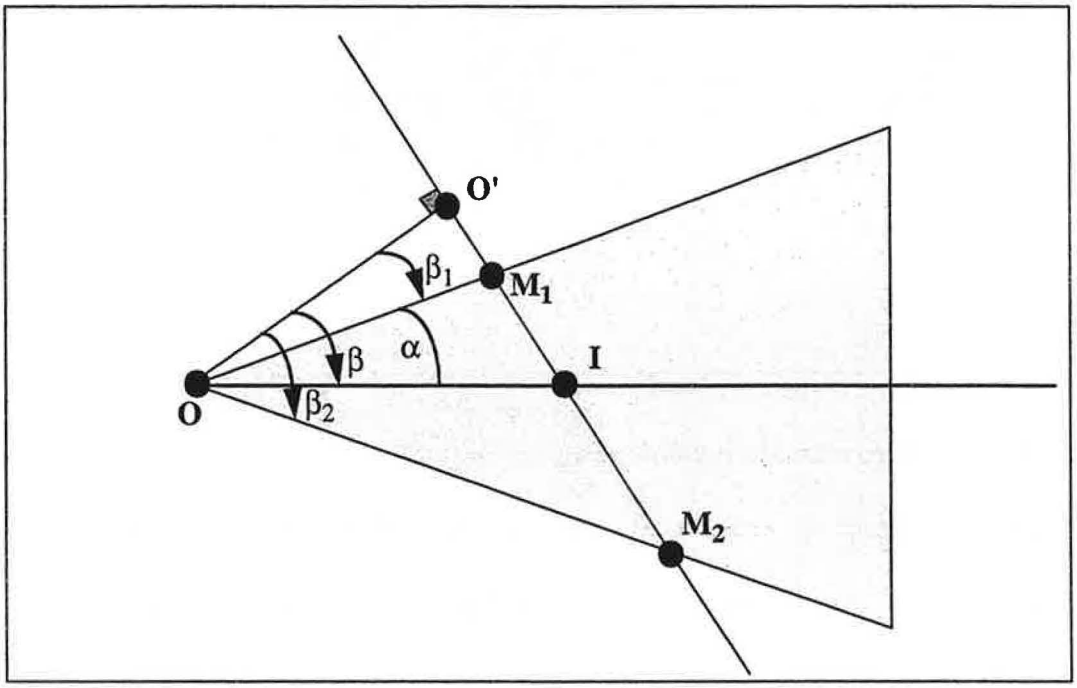
Il ne faut pas perdre de vue que nous cherchons à borner supérieurement la taille de la projection. Celle-ci peut l'être en remplaçant le cône de projection de la calotte par un cône de révolution englobant le premier. Pour la même raison que précédemment, ces cônes sont identiques à une rotation autour de (OS) près. Reprenons alors le calcul de la projection d'une sphère sur le plan (P) pour calculer le grand axe de l'ellipse, qui est bien le maximum des dimensions de la projection de la sphère.

II-5-2-2) Le calcul du grand axe

Il a été démontré précédemment (voir "Remarque importante:", page 58)que la projection orthogonale de l'observateur sur le plan de projection se trouve sur la droite support du grand axe de l'ellipse. Ainsi, on peut calculer la longueur maximale de

l'intersection du cône et du plan. On prendra comme notations celles de la figure 37, page 68.

Figure 37. **Notations pour le calcul du grand axe**



De la figure 37, page 68, on déduit:

$$\overline{M_1M_2} = \overline{O'M_2} - \overline{O'M_1} = OO' \cdot (\tan\beta_2 - \tan\beta_1)$$

d'où

$$M_1M_2 = OO' \cdot (\tan\beta_2 - \tan\beta_1) = OO' \cdot (\tan(\beta + \alpha) - \tan(\beta - \alpha))$$

Développons maintenant la partie trigonométrique:

$$\tan(\beta+\alpha)-\tan(\beta-\alpha) = \frac{\tan\beta+\tan\alpha}{1-(\tan\beta\cdot\tan\alpha)} - \frac{\tan\beta-\tan\alpha}{1+(\tan\beta\cdot\tan\alpha)}$$

$$= \frac{(\tan\beta+\tan\alpha)\cdot(1+(\tan\beta\cdot\tan\alpha))-((\tan\beta-\tan\alpha)\cdot(1-(\tan\beta\cdot\tan\alpha)))}{1-(\tan\beta\cdot\tan\alpha)^2}$$

$$= \frac{2\cdot\tan\alpha\cdot(1+(\tan\beta)^2)}{1-(\tan\alpha\cdot\tan\beta)^2}$$

d'où enfin:

$$M_1M_2 = OO'\cdot\frac{2\cdot\tan\alpha\cdot(1+(\tan\beta)^2)}{1-(\tan\alpha\cdot\tan\beta)^2}$$

Cette formule permet de calculer rapidement la longueur du grand axe de la projection perspective d'une sphère. Elle ne peut cependant pas être utile dans les calculs précédents où la mesure des translations de l'objet nécessite les points extrema et non seulement cette longueur.

II-5-2-3) Comment maximaliser la longueur du grand axe

Reprenons la formule précédente. Elle dépend de deux paramètre α et β. β est l'angle entre le vecteur de visée et l'axe du cône de projection. α est l'angle d'ouverture de ce cône.

Considérons l'ensemble des points obtenus par rotation autour de (OS) d'un point de la sphère. Puisque, du point de vue de la sphère et de l'observateur, la situation est conservée par n'importe laquelle de ces rotations, les cônes de projection des calottes

sphériques associées à ces points sont identiques: α est une constante. Posons alors ($\tan\alpha = T$) et calculons les variations de la longueur du grand axe en fonction de β.

$$M_1M_2 = f_\alpha(\beta) = K \cdot \frac{(1 + (\tan\beta)^2)}{1 - (T \cdot \tan\beta)^2}$$

soit en posant $t = \tan\beta$:

$$F_\alpha(t) = \frac{f_\alpha(\beta)}{K} = \frac{(1 + t^2)}{1 - (T \cdot t)^2}$$

d'où:

$$\frac{\partial F}{\partial t} = \frac{2t \cdot (1 + T^2)}{[1 - (T \cdot t)^2]^2} \geq 0$$

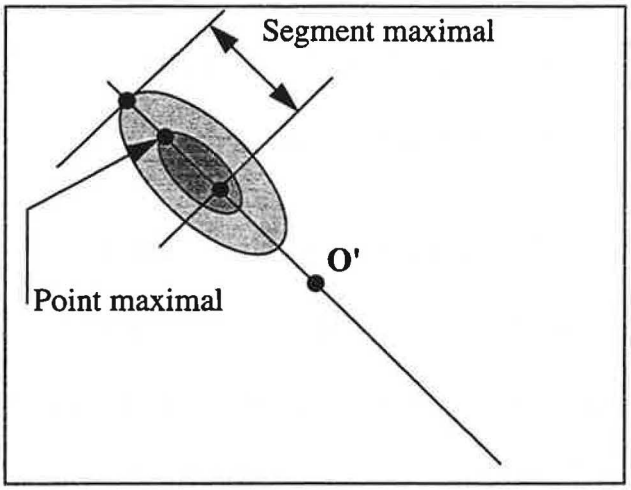
Le cas $(1-(T.t)^2 = 0)$ n'est jamais atteint. Il correspond en effet à une projection de taille infinie, ce qui est impossible puisque nous nous sommes placés dans le cas où la projection de la sphère englobante toute entière est une ellipse, c'est à dire ne comporte pas de point à l'infini. La calotte, qui est un sous-ensemble de la sphère ne peut donc pas avoir de point à l'infini.

La taille de la projection est en conséquence une fonction croissante de l'angle entre le vecteur de visée et la position du point sur la sphère. Ainsi, on bornera supérieurement le déplacement des points d'une couronne par celui du point d'angle β le plus grand.

Puisque l'on sait que la projection O' de l'observateur se trouve sur le support du grand axe de l'ellipse, le point de plus grand angle β se trouve sur le segment joignant le

centre de celle-ci et l'extrémité du grand axe la plus éloignée, que l'on nommera «segment maximal»

Figure 38. La projection d'une couronne et son point maximal.



II-5-3) La mesure de l'impact des rotations

II-5-3-1) Le segment maximal

Malgré nos efforts, nous ne sommes pas parvenu à trouver le ou les points maximaux vis à vis des positions des couronnes, c'est à dire des points qui auraient un déplacement supérieur à tous les autres. En effet, en se déplaçant à la surface de la sphère, les influences de la distance à l'observateur, de l'angle par rapport au vecteur de visée, de la «direction» de la rotation ou encore de la longueur de l'arc de celle-ci deviennent très complexes et chaque rotation est un cas particulier.

Nous pouvons simplement déduire du calcul du paragraphe précédent que les vitesses de déplacement dans l'espace de l'image sont maximales sur le segment maximal. Ainsi, une évaluation de la mesure du déplacement induit par les rotations peut

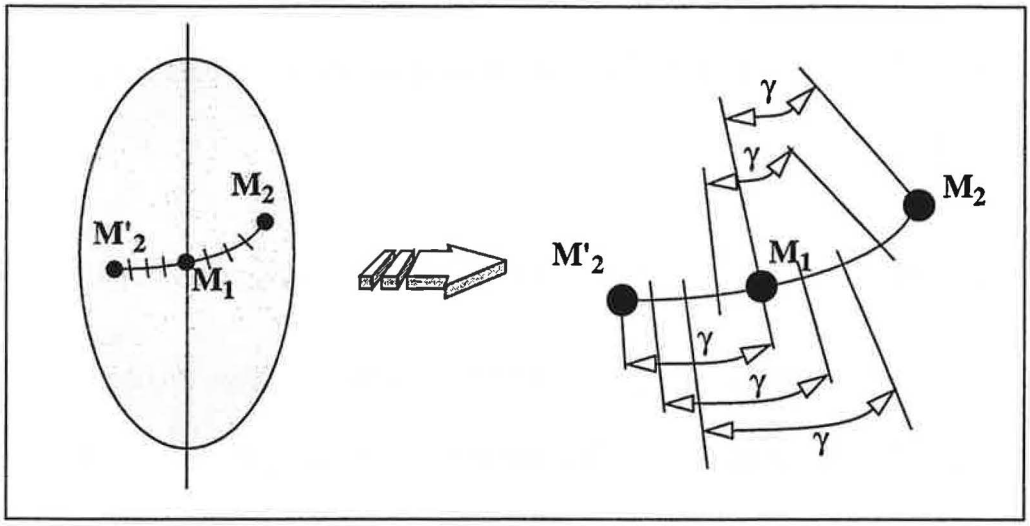
être obtenue en discrétisant le segment maximal et en mesurant les mouvements autour des points ainsi obtenus.

Afin de calculer ces mouvements, on commence par calculer quels sont les points de la sphère qui se projettent en chaque pixel du segment maximal discrétisé: ce sont les intersections entre la sphère et les rayons passant par les pixels.

Lors de l'animation, ces points restent solidaires de la sphère et se déplacent dans l'espace 3D, suite aux mouvements à la fois de l'objet et de l'observateur. La résultante de ces rotations d'axes passant par le centre la sphère est une rotation d'axe inconnu passant lui aussi par ce centre. A la fin d'une période d'animation, on connaît la position finale de chaque point du segment maximal. Or, on sait que c'est autour de ces points que les mouvements sont maximaux, mais on ne peut, pour les mêmes raisons que précédemment, dire où exactement la rotation induit un déplacement maximal dans l'image.

Ainsi, nous allons de la même manière discrétiser l'arc centré sur le point du segment maximal et de demi-angle d'ouverture l'angle de la rotation du point correspondant sur la sphère. On nommera cet arc «arc de rotation» et on le discrétisera à angle constant avec un nombre total de points soit fixe, soit proportionnel à la longueur en pixels de celui-ci. En projetant ensuite les points de la sphère ainsi obtenus dans l'espace de l'image, on peut calculer la longueur apparente dans l'espace image de chaque doublet de points séparés par l'angle de la rotation (voir figure 39, page 73).

Figure 39. **La discrétisation de l'arc de rotation**



II-5-3-2) La définition de la rotation résultante

Un changement de repère entre deux instants de l'animation nous permet de considérer l'observateur comme fixe. Seule la sphère est mobile dans la suite de ce chapitre.

On connaît deux positions M_1 et M_2 à deux instants donnés d'un point de la surface de la sphère. On peut calculer M_3 comme le transformé de M_2 par la succession de rotations. Le plan (M_1, M_2, M_3) est celui de l'arc de rotation de M_1. L'axe de la rotation est perpendiculaire à ce plan et passe par le centre de la sphère. Le centre de l'arc de rotation est donc la projection orthogonale du centre de la sphère sur le plan (M_1, M_2, M_3).

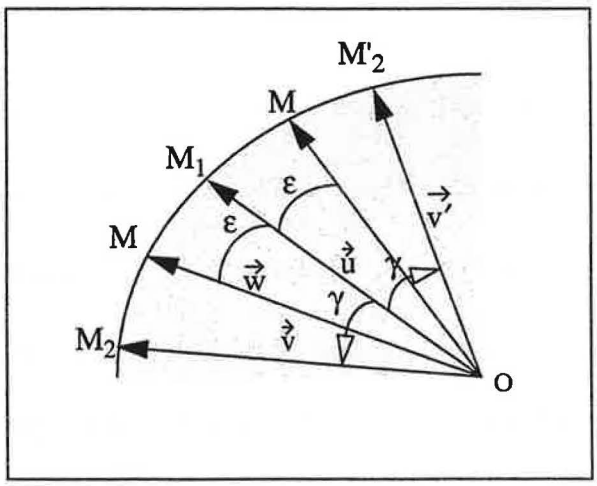
Remarque: ce plan n'est pas défini dans deux cas. Si la rotation résultante est l'identité, le cas est facile à traiter. Si M_1 est identique à M_3 après deux rotations d'angle π, on définit alors M_3 comme le résultat d'une demi-rotation de M_1.

II-5-3-3) La discrétisation des arcs de déplacement

On cherche à discrétiser l'arc centré sur M_1 et de demi-angle au centre l'angle entre les deux positions.

Définissons alors le vecteur $\vec{u}$ comme le vecteur joignant le centre de rotation O et le point M_1 et $\vec{v}$ celui joignant O et M_2. On définit également $\vec{v'}$ comme joignant O au symétrique M'_2 de M_2 par rapport à M_1. On discrétise alors l'arc $M_2M'_2$ à pas constant sur l'angle par rapport à OM_1. Il faut alors calculer la position d'un point M quelconque sur cet arc, connaissant la valeur absolue de son angle ε: deux points sont alors solution (voir figure 40, page 74).

Figure 40. Les points des arcs de déplacement.



Posons alors:

$$\vec{w} = \alpha \cdot \vec{u} + \beta \cdot \vec{v}$$

Si on note R le rayon de l'arc, les positions du point M peuvent être définies de la manière suivante:

$$\begin{cases} (\vec{w}^2 = R^2) \Leftrightarrow \text{M appartient à l'arc de cercle} \\ (\vec{u} \cdot \vec{w} = R^2 \cdot \cos\varepsilon) \Leftrightarrow \text{Son angle est } \varepsilon \end{cases}$$

Calculons alors les expressions ci-dessus en fonction de α et β:

$$\begin{cases} \vec{w}^2 = (\alpha \cdot \vec{u} + \beta \cdot \vec{v})^2 = \alpha^2 \cdot \vec{u}^2 + \beta^2 \cdot \vec{v}^2 + 2\alpha\beta \cdot \vec{u} \cdot \vec{v} \\ \vec{u} \cdot \vec{w} = \vec{u} \cdot (\alpha \cdot \vec{u} + \beta \cdot \vec{v}) = \alpha \cdot \vec{u}^2 + \beta \cdot \vec{u} \cdot \vec{v} \end{cases}$$

soit:

$$\begin{cases} \vec{w}^2 = R^2 \cdot (\alpha^2 + \beta^2 + 2\alpha\beta \cdot \cos\gamma) = R^2 \\ \vec{u} \cdot \vec{w} = R^2 \cdot (\alpha + \beta \cdot \cos\gamma) = R^2 \cdot \cos\varepsilon \end{cases}$$

soit le système en α et β à résoudre:

$$\begin{cases} \alpha^2 + 2\alpha\beta \cdot \cos\gamma + (\beta^2 - 1) = 0 \\ \alpha + \beta \cdot \cos\gamma = \cos\varepsilon \end{cases}$$

Procédons par substitution et remplaçons α par sa valeur en fonction de β dans l'équation du second degré.

$$\alpha = \cos\varepsilon - \beta \cdot \cos\gamma$$

$$\Rightarrow (\cos\varepsilon - \beta \cdot \cos\gamma)^2 + 2(\cos\varepsilon - \beta \cdot \cos\gamma)\beta \cdot \cos\gamma + (\beta^2 - 1) = 0$$

$$\Rightarrow \beta^2 \cdot [\cos\gamma^2 - 2\cos\gamma^2 + 1] + \beta \cdot [2\cos\gamma \cdot \cos\varepsilon - 2\cos\gamma \cdot \cos\varepsilon] + [\cos\varepsilon^2 - 1] = 0$$

$$\Rightarrow \beta^2 \cdot [1 - \cos\gamma^2] = [1 - \cos\varepsilon^2]$$

d'où finalement:

(EQ 2)

$$\vec{w} = \alpha \cdot \vec{u} + \beta \cdot \vec{v}$$

$$\beta = \pm\frac{\sin\varepsilon}{\sin\gamma} \qquad \alpha = \cos\varepsilon - \beta \cdot \cos\gamma$$

$$\text{avec} \qquad \cos\gamma = \frac{\vec{u} \cdot \vec{v}}{\vec{u}^2} \qquad \varepsilon \text{ donné}$$

Remarque d'implémentation:

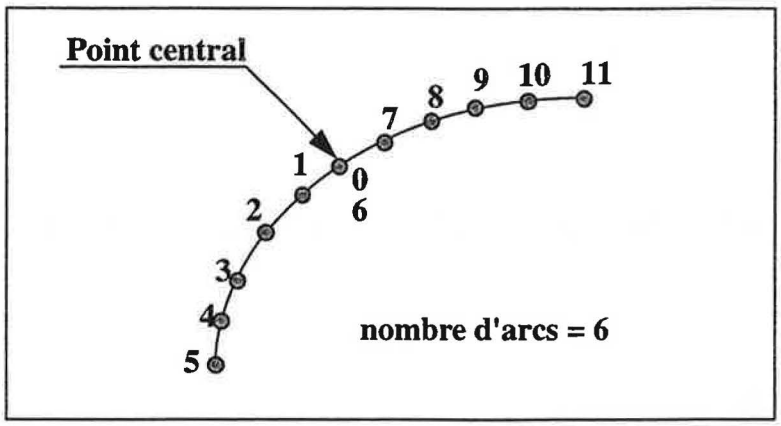
les formules de (EQ 2) semblent comporter de nombreux calculs trigonométriques qui sont coûteux en temps de calcul (voir Annexe B, page 119). Dans la pratique, les angles peuvent être calculés de manière incrémentale. Posons δ l'incrément des angle ε. On a alors:

$$\varepsilon' = \varepsilon + \delta$$
$$\cos\varepsilon' = \cos(\varepsilon + \delta) = \cos\varepsilon \times \cos\delta - \sin\varepsilon \times \sin\delta$$
$$\sin\varepsilon' = \sin(\varepsilon + \delta) = \sin\varepsilon \times \cos\delta + \cos\varepsilon \times \sin\delta$$

On calcule cos δ et sin δ une fois pour toute. Les calculs suivants ne demandent que quatre multiplications et deux additions.

Les points seront stockés dans un tableau dans l'ordre de la figure 41, page 77.

Figure 41. **La numérotation des points**



L'arc numéro i (i=0..5) joint alors les points i et 2*nb_arcs-1-i.

II-5-3-4) La mesure des rotations

Pour chaque point du segment maximal S_{max}, on est ainsi capable de calculer la longueur $L_{doublet}$ de la projection de chaque doublet de point de l'arc de rotation Arc_{rot}. On définira alors la mesure des rotations par:

$$D_{rot} = \underset{(S_{max})}{\text{Max}}\{\underset{(Arc_{rot})}{\text{Max}}\{L_{doublet}\}\}$$

II-6) Performances

Nous tentons maintenant d'évaluer le coût temporel de la mesure du mouvement. Pour cela, nous décomposerons les opérations effectuées dans notre implémentation en trois catégories selon le temps qu'elles nécessitent: les opérations d'addition (+,-,*), de division (division entre deux réels) et les opération complexes (racines carrées, calcul

trigonométrique). Chaque procédure peut alors être représentée par un triplet (a,b,c) qui est le nombre de chacune de ces opérations.

L'ellipse:

Le calcul complet de l'ellipse demande 170 additions et seulement 7 divisions et 4 opérations complexes.

Mesure des rotations

Le calcul des points résultant des divers mouvements est (61,7,7). Celui du centre de rotation demande 34 additions et seulement 1 division (34,1,0). Celui enfin des points de segmentation de l'arc de rotation nécessite (29 + 43*N) additions, où N est le nombre de segments désirés plus 3 divisions et 4 opérations complexes (29+43*n,3,4).

Chaque point du segment maximal impose en conséquence un coût temporel de (134+43*N, 11, 11). Soit un coût global de la mesure des rotations de M*(134+43*N, 11, 11), où M est le nombre de points du segment maximal. On remarquera que N est de l'ordre de M, soit proche de dix dans la plupart des cas.

A titre d'exemple, le traitement de 100 000 objets sur une station Silicon Graphics (Indigo 2, R4400 à 250 MHz) prend 5 secondes 450 milli-secondes soit environ $5.\ 10^{-5}$ seconde par objet alors que le calcul de l'image d'une centaine de ces objets nécessite de l'ordre de la dizaine de secondes. Le coût de la mesure est en conséquence très faible relativement à celui du calcul d'image.

II-7) Conclusions

Suite aux paragraphes précédents, nous sommes capables de mesurer l'impact des différents mouvements d'un objet relativement à un observateur, cela en terme de déplacement des points de sa projection dans l'espace de l'image. On définira le mouvement de l'objet comme étant le maximum des mouvements induits en translations ou en rotations.

Dans le chapitre suivant, nous allons maintenant tenter d'utiliser cette connaissance des modifications de l'apparence des objets dans un rendu d'animations de synthèse utilisant la notion de cohérence temporelle.

CHAPITRE III

Mais aussi de l'animation

III-1) Introduction

L'objectif de ce travail est la production d'animations de synthèse répondant aux critères suivants:

- Réduction des temps de calcul.
- Pas de limitation des mouvements de l'observateur.
- Aucune connaissance a-priori de l'animation n'est nécessaire.
- Réalisme du résultat.

Afin de diminuer les temps de calcul, nous avons choisi d'utiliser la notion de cohérence temporelle. Cette cohérence sera évaluée grâce à la mesure du mouvement décrite au chapitre précédent et permettra la segmentation de la base de données des objets à visualiser en deux catégories:

- Les objets mobiles qu'il faudra calculer à chaque image.
- Les objets dits «du décor», c'est à dire les objets dont l'apparence, «peu modifiée» au cours d'une partie de l'animation, ne demande pas un calcul

complet à chaque image.

L'animation est ensuite reconstruite à partir des images des deux parties de la base de données en respectant le critère de réalisme, c'est à dire que le résultat obtenu doit être équivalent à celui qui le serait par un calcul plus classique (image par image) de l'animation.

Nous allons maintenant présenter dans le détail ces étapes de notre logiciel, avant de donner quelques exemples des résultats obtenus, aussi bien du point de vue de la qualité de l'image produite que du temps de calcul économisé.

III-2) La segmentation de la base de données

III-2-1) Introduction

La segmentation de la base de données initiale en plusieurs bases de données intermédiaires est le coeur du logiciel. C'est en effet celle-ci qui va permettre de tirer profit de la cohérence temporelle présente dans l'animation.

Son objectif est de détecter parmi les objets qui lui sont proposés ceux qui sont susceptibles de provoquer une modification visible dans l'animation. En cas d'impossibilité de fournir une réponse certaine à cette question pour un objet, celui-ci sera déclaré comme mobile. Il vaut mieux en effet recalculer inutilement un objet que fournir une animation contenant une erreur visuellement perceptible.

La fonction de segmentation a pour paramètres d'entrée la liste des objets, caractérisés par leur sphère englobante et leurs vitesses initiales. On suppose ici que ces

informations sont connues. Le format des fichiers de données et leur exploitation sont décrits dans l'Annexe A, page 117

A partir de ces données, on peut prédire le comportement des objets de la scène en prenant comme hypothèse la conservation des vitesses. On connaît la position de départ et l'on suppose une position d'arrivée après un certain nombre d'images.

La fonction de segmentation peut avoir pour résultat quatre valeurs différentes:

- IMMOBILE
- CLIPPE
- LENT
- RAPIDE

Nous allons maintenant exposer les cas aboutissant à chacun de ces quatre résultats.

III-2-2) Les objets immobiles

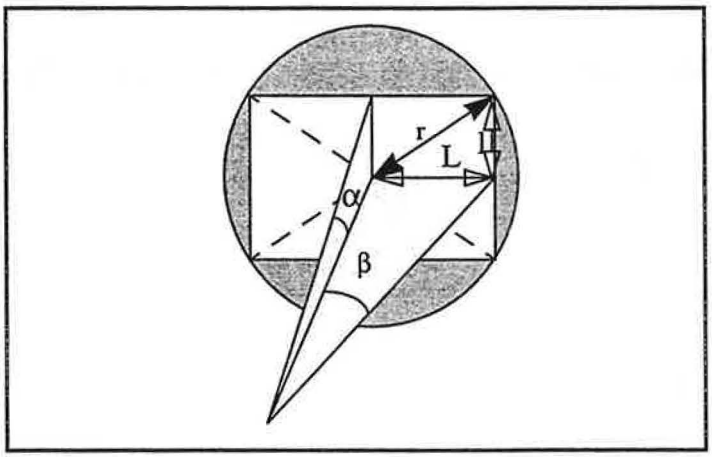
Un objet est marqué IMMOBILE si sa vitesse relative à l'observateur est nulle. Ce marquage permet de traiter très rapidement la plupart des objets dans le cas où l'observateur est fixe.

III-2-3) Les objets en dehors du champ de la caméra

Le premier test à faire sur les objets est de vérifier qu'ils apparaîtront dans les images produites, c'est à dire qu'ils se trouvent dans le champ de la caméra.

Une caméra est généralement décrite par sa position, ses angles d'ouverture horizontaux et verticaux et son point de visée. Dans un souci de simplification des calculs, la pyramide de visualisation, qui est à base rectangulaire, sera remplacée par le cône qui l'englobe.

Figure 42. **Le cône de visualisation**



On calcule aisément l'angle χ du cône en remarquant qu'un plan perpendiculaire à l'axe de visée coupe la pyramide en un rectangle et le cône en un cercle. Le rayon r de ce cercle est l'hypoténuse d'un triangle dont les deux autres côtés sont les demi-côtés du rectangle (l et L). Ces derniers sont donnés par la distance D du plan et les demi-angles d'ouverture de la pyramide α et β:

$$l = d \times \tan\alpha$$
$$L = d \times \tan\beta$$
$$r = d \times \tan\chi$$

avec $r^2 = l^2 + L^2$ soit

$$\tan\chi = \sqrt{\tan\alpha^2 + \tan\beta^2}$$

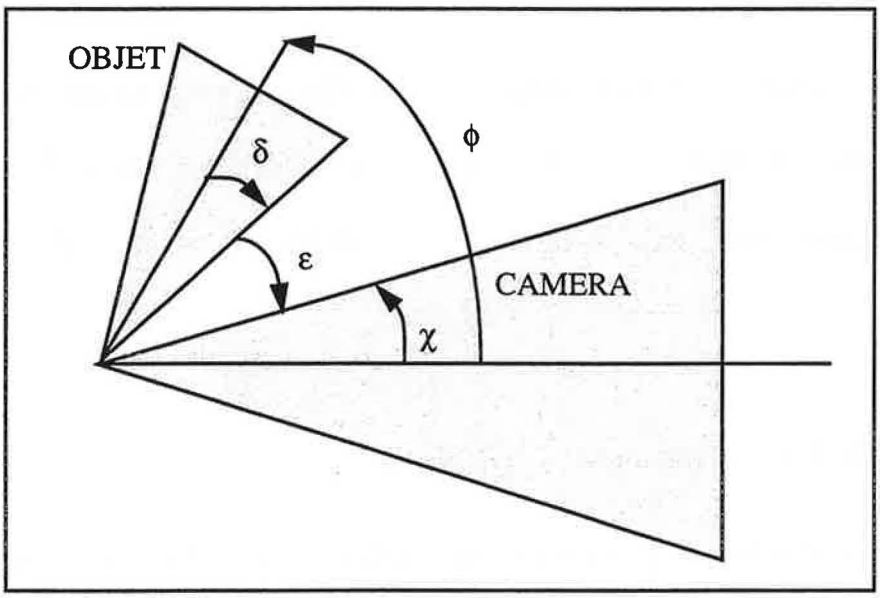
On peut également calculer très facilement l'angle δ du cône tangent à la sphère englobante de l'objet (voir Figure 30, "Le contour d'une sphère", page 55).

$$\sin\delta = \frac{R}{D} \qquad \tan\delta = \frac{R}{\sqrt{D^2 + R^2}}$$

si R est le rayon de la sphère englobante et D la distance entre la caméra et le centre de la sphère.

Pour savoir si l'objet se trouve dans le champ de la caméra, il suffit alors de savoir quel est l'angle qui sépare ces deux cônes comme dans la figure ci-dessous:

Figure 43. Les angles de clipping



L'angle qui sépare les deux cônes est donc $\varepsilon = \phi - (\delta+\chi)$ où ϕ est l'angle entre le vecteur de visée de la caméra et le vecteur caméra-objet.

L'objet est en dehors du champ de la caméra si ε est positif.

Dans la pratique, comme seul le signe de ε nous intéresse, on calculera plutôt sa tangente. Celle-ci est plus rapide à calculer à partir des tangentes des autres angles, car on n'a alors pas de fonction réciproque de fonction trigonométrique à calculer:

$$\tan\varepsilon = \tan(\phi - (\delta + \chi)) = \frac{\tan\phi - \tan(\delta + \chi)}{1 + \tan\phi \cdot \tan(\delta + \chi)}$$

$$\tan\varepsilon = \frac{\tan\phi \cdot (1 - \tan\delta \cdot \tan\chi) - (\tan\delta + \tan\chi)}{(1 - \tan\delta \cdot \tan\chi) + \tan\phi \cdot (\tan\delta + \tan\chi)}$$

La tangente de ϕ est calculée en remarquant que la tangente de l'angle entre deux vecteurs est le rapport de la norme de leur produit vectoriel par leur produit scalaire.

On se reportera à l'Annexe B, page 119 pour le détail des temps de calcul.

Si un objet se trouve en dehors du champ de la caméra pour ses deux positions extrêmes, il y a de fortes chances pour qu'il ne traverse pas ce champ au cours de l'animation et n'intervienne jamais dans le calcul réel des images. Il est alors marqué CLIPPE.

III-2-4) Les objets quasi-immobiles ou objets lents

Ces objets sont ceux dont l'apparence est «peu modifiée», c'est à dire ceux dont le mouvement, au sens du chapitre précédent, est inférieur à un certain seuil. Ils sont marqués LENT. Dans la suite de ce travail, on nommera ce seuil «seuil de mouvement»

Ce seuil, qui est une distance dans l'espace de l'image, peut être défini par défaut, par l'utilisateur ou encore par l'application elle-même. Il agit comme une mesure de la qualité des images produites en autorisant des déplacements plus ou moins grands des objets dans l'espace image.

Une personne qui souhaite réaliser une prévisualisation d'une animation fixera un seuil élevé afin de réduire le temps des calculs. Ce seuil lui permet néanmoins de conserver un certain contrôle de l'erreur qu'il commet.

Une application qui est commandée par le temps dont elle dispose pour calculer une image pourra quant à elle ajuster le seuil à la complexité de la scène à visualiser. Elle pourra par exemple multiplier le seuil précédent par Q (avec Q>1) si elle n'a pas eu le temps de calculer complètement les images précédentes ou au contraire le diviser si elle a auparavant sacrifié la qualité des images aux temps de calculs et qu'il lui est resté en définitif du temps inemployé.

III-2-5) Les objets mobiles ou objets rapides

Les objets dont le déplacement est supérieur au seuil précédemment décrit sont marqués RAPIDE.

De manière générale, tous les objets qui n'ont pas été marqués précédemment sont également marqués RAPIDE. Ceci concerne particulièrement les cas où la mesure du mouvement n'a pas pu être faite comme dans le cas où l'observateur se trouve à l'intérieur de la sphère englobante ou celui où la conique de projection de la sphère est une hyperbole. On ne peut en effet fournir d'information sur l'évolution de l'apparence des objets dans ces cas précis, à moins bien sur que les objets aient déjà été marqués IMMOBILE.

III-3) Le calcul de l'animation

III-3-1) Le processus de prédiction-vérification

Le calcul de l'animation débute par un processus de prédiction basé sur la segmentation précédemment décrite. Le pari est fait que les objets conserveront une vitesse proche de leur vitesse initiale pendant une certaine période de temps, nommée «période de la prédiction». Si l'on ne connaît pas l'animation à l'avance, on peut en conséquence calculer une position finale prédite qui correspondrait au déplacement de l'objet à vitesse constante au cours de la période de prédiction. Cette dernière correspond à un nombre d'images produites que l'on nommera par la suite «taille de la prédiction». Dans le cas où l'animation est connue a priori, on utilisera la position réelle de l'objet à la fin de la période de prédiction et l'on prendra le pari que le déplacement entre les deux points est effectué à vitesse constante.

La segmentation de la base de données des objets peut être alors réalisée entre la position initiale et cette position finale prédite. Le seuil de mouvement variera en fonction de la qualité du résultat souhaité comme expliqué dans le processus de segmentation.

Les objets marqués IMMOBILE, CLIPPE ou LENT sont regroupés car ce sont les objets dont il sera inutile de calculer l'apparence pour chaque image durant la période de prédiction. On les nomme objets immobiles au sens du seuil de mouvement, ou plus simplement objets du décor.

Il s'agira par la suite de vérifier la validité de cette prédiction afin d'obtenir une animation valide:

•les objets IMMOBILE et CLIPPE doivent garder leur marquage.

•on calcule pour chaque objet LENT quel est son mouvement entre sa position initiale et sa position actuelle réelle. Ce mouvement doit rester inférieur au seuil fixé.

III-3-2) Le rendu de l'animation

La prédiction permet la constitution de deux bases de données. La première est constituée par les objets rapides et regroupe les objets dont il faut calculer l'image à chaque pas de l'animation. La seconde est la réunion des objets immobiles au sens du seuil de mouvement.

On distingue alors deux méthodes de construction de l'animation: l'une qui privilégie la diminution des temps de calcul et l'autre qui tente de s'approcher le plus possible du rendu qui serait obtenu en calculant l'animation de manière classique, image par image.

III-3-2-1) Le rendu simplifié

Le rendu simplifié est utilisé soit dans les applications où la qualité n'est pas un impératif comme la pré-visualisation d'une animation, soit dans celles où les temps de calcul alloués imposent une qualité moindre, comme les logiciels de visualisation temps-réel.

Après la phase de prédiction, il existe deux bases de données distinctes d'objets. On fournit alors ces dernières à un logiciel de rendu externe en les traduisant au format de scènes imposé par celui-ci. Cette indépendance vis à vis du logiciel de rendu permet

de s'adapter très rapidement à un nouvel environnement de travail: seul le traducteur est alors à modifier.

Au cours de la validation de nos résultats, nous avons par exemple utilisé l'environnement Illumines [BEIG89], [ROEL93] développé à l'Ecole des Mines de Saint-Etienne. Celui-ci nous impose un format de scène «Castor», qui est un fichier ASCII à philosophie CSG aisément manipulable.

Au début de la période de prédiction, on calcule une image de chacune des deux bases de données. On se servira par la suite de l'image des objets immobiles au sens du seuil de mouvement comme d'un décor dans lequel évoluent les objets rapides.

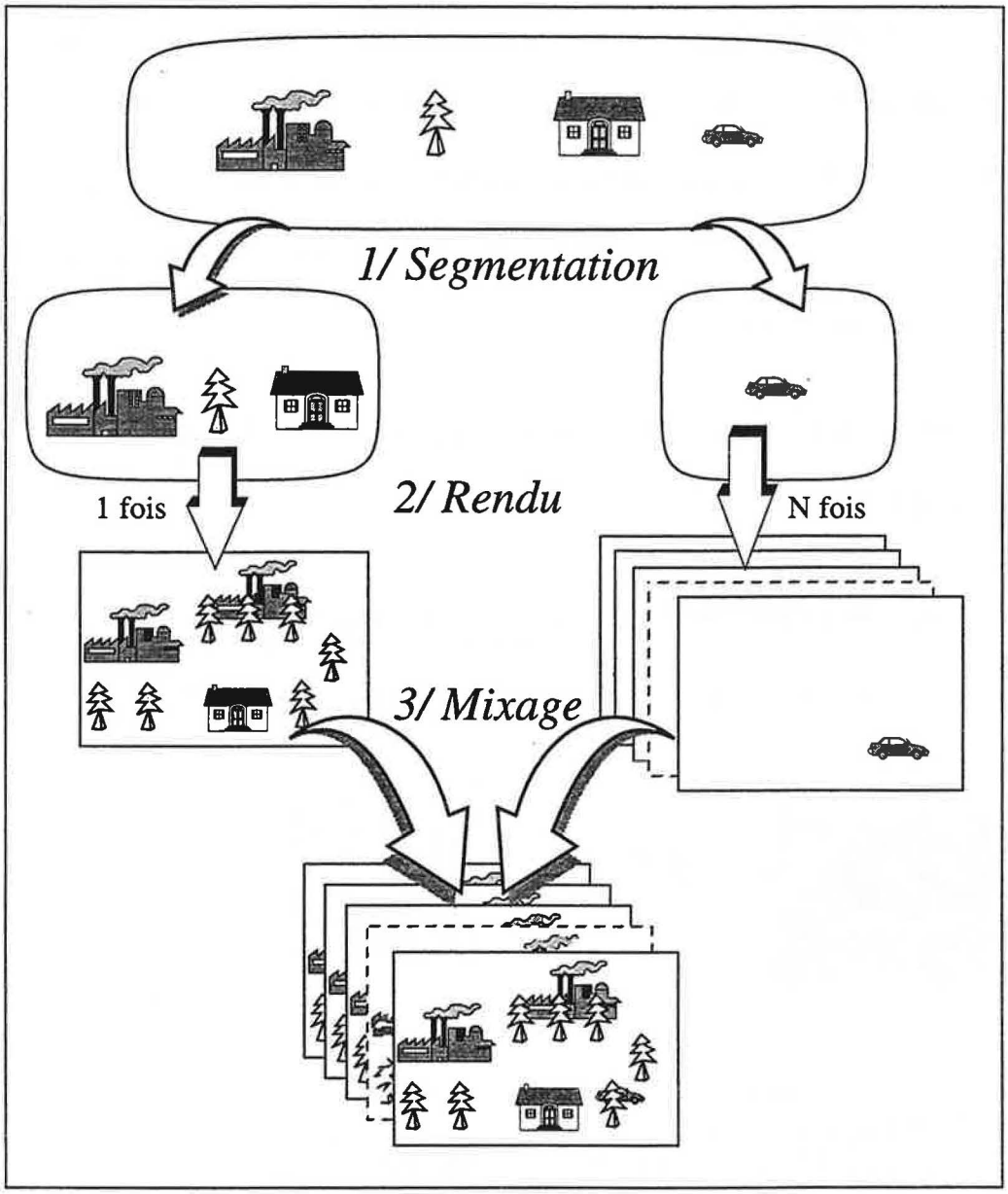
Une contrainte est en conséquence imposée au logiciel de rendu. En effet, celui-ci doit produire non seulement une image de la scène mais également sa carte de profondeur, qui est utilisée dans la phase de reconstruction de l'animation. Cette contrainte n'est cependant pas très forte puisque la plupart des logiciels calculent cette profondeur, il suffit de penser à la sauvegarder! On appellera par la suite «décor», l'union de l'image et de la carte de profondeur.

Pour la reconstruction de chaque image de l'animation, on calculera ainsi une image des objets rapides que l'on mixera ensuite avec le décor par la technique classique du tampon de profondeur([CATM74]): on prendra pour couleur finale du pixel la couleur du pixel le plus proche entre celui du décor et celui des objets dits rapides.

En cas de détection d'erreur au cours de la phase de vérification, le décor n'est plus valide. Il suffit alors d'en recalculer un nouveau et de commencer une nouvelle période de prédiction.

Si l'on reprend l'exemple de la voiture (voir Figure 1, "Le calcul classique d'une animation", page 8) le calcul de l'animation de synthèse se déroule de la manière suivante:

Figure 44. Le fonctionnement général du rendu simplifié



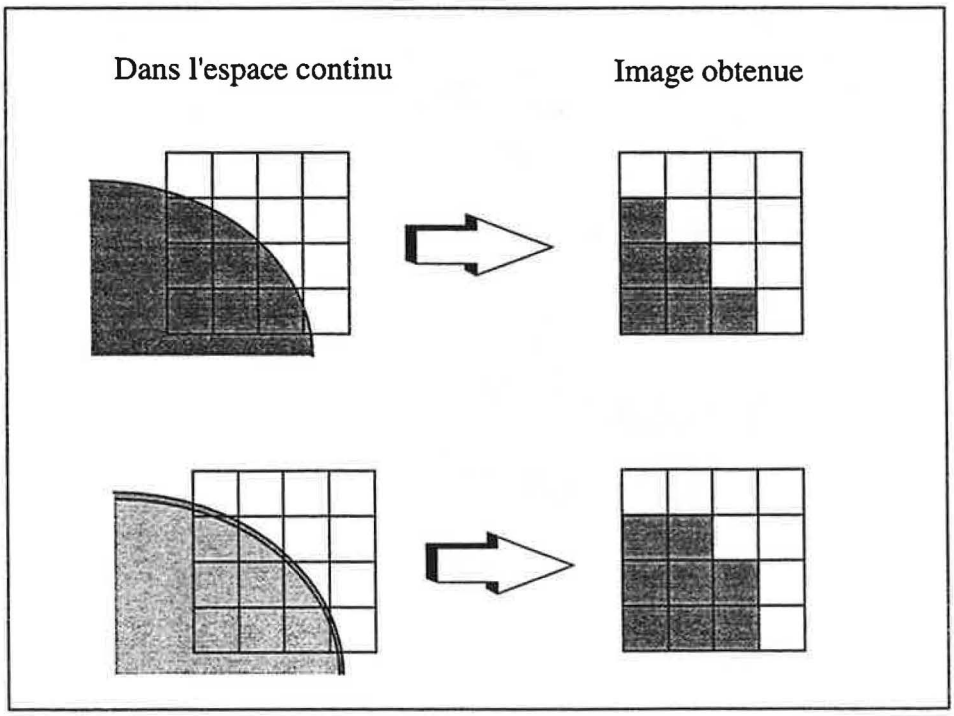
Le rendu simplifié a l'avantage de l'efficacité et trouve son utilité dans les applications où le temps est le point critique. Cet algorithme trouve néanmoins ses limites dans la qualité de l'animation produite. En effet, si celle-ci est cohérente à l'intérieur de

chaque période de prédiction, une discontinuité qui peut être visuellement perceptible existe à la transition entre ces périodes.

Comme il est naturel de le penser, cette discontinuité est fonction du seuil de mouvement fixé, puisqu'il correspond au déplacement que l'on a autorisé pour l'objet dans l'espace image. Cependant, on ne peut pas contrôler ce saut en dessous de la taille d'un pixel, quel que soit le seuil que l'on fixe. L'image produite est en effet discrète et, dans un logiciel de tracé de rayons classique, seul le rayon lancé au centre du pixel est pris en compte, un déplacement infime de l'objet peut donc l'amener à couper le rayon. C'est le problème classique de l'aliasage.

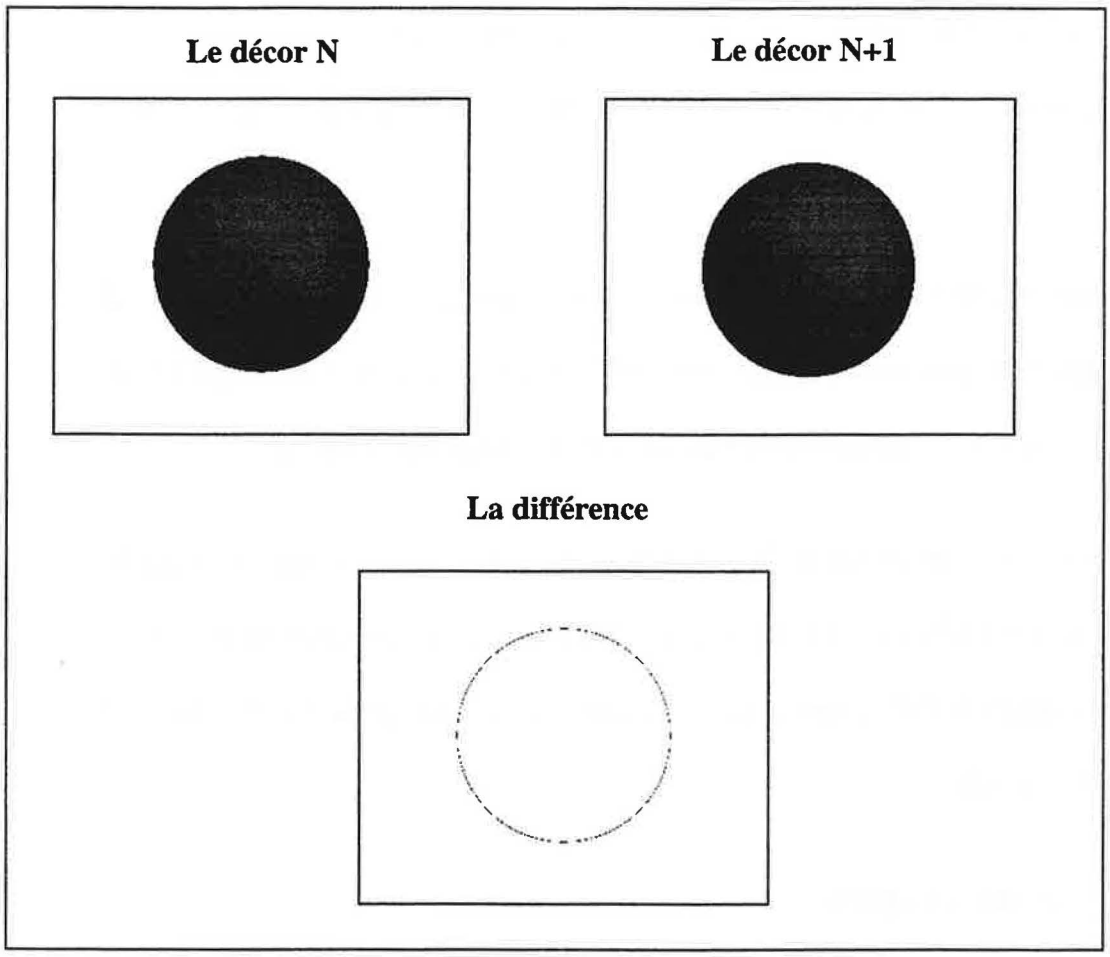
Dans l'exemple ci-dessous par exemple, un déplacement infime de l'objet entraîne la modification de deux pixels.

Figure 45. **Déplacement infime, grandes conséquences.**



Dans le cas d'un observateur qui se rapproche d'une sphère, c'est toute le contour apparent de celle-ci qui grandit et l'on voit brusquement surgir une nouvelle couronne à l'ellipse existante.

Figure 46. **Zoom sur une sphère**



Il est bien entendu possible de réduire ces discontinuités en utilisant des techniques moins «brutales» que le tampon de profondeur classique comme le A-buffer proposé par Lauren Carpenter[CARP84]. Cependant. l'oeil est très sensible au mouvement simultané de zones de l'image et les transitions entre les différentes périodes de prédiction peuvent être néanmoins perceptibles.

Ainsi, nous allons maintenant proposer un algorithme de traitement du décor afin de rendre l'utilisation du processus de prédiction-vérification imperceptible.

III-3-2-2) Le traitement du décor

Ce traitement n'est utile que dans le cas où l'on souhaite une animation équivalente à celle qui serait produite par un logiciel de calcul image par image. En conséquence, on prendra comme seuil du mouvement autorisé la longueur d'un pixel. Ainsi, les seules modifications possibles sont celles dues à la discrétisation de l'image, c'est à dire un déplacement d'un pixel à l'un ou plusieurs de ses voisins comme dans la Figure 45. , page 92.

Puisque l'objectif visé est le calcul rapide d'animation, nous conservons ici comme méthode de mixage des différentes parties de l'image la technique du tampon de profondeur. Cette technique est en effet la plus efficace en terme de rapidité.

Deux cas se présentent alors. S'il s'agit du calcul d'une animation qui n'est visualisée qu'une fois achevée, on peut se permettre d'effectuer un post-traitement. Si l'on est par contre en temps réel et que les images sont affichées aussitôt produites, le traitement doit être fait à la volée.

Le cas du post-traitement

Afin de permettre un post-traitement, le mixage des images n'est pas fait immédiatement mais est reporté au début de la période de prédiction suivante. A l'image n, nous avons calculé un décor, que l'on notera décor(n), qui a servi pour p images. A l'instant (n+p), on calcule un nouveau décor(n+p) pour les images suivantes. Ce décor est l'image d'une nouvelle base de données obtenue par une nouvelle segmentation. Ce ne sont donc pas, a priori, les mêmes objets que ceux du décor précédent.

Afin de connaître l'apparence réelle des objets de décor(n) à cette jonction, on calcule leur image à l'instant (n+p). Il est à remarquer que, très souvent, la liste des objets du décor varie très peu entre deux périodes de prédiction. Il est en conséquence avantageux de calculer le décor(n+p) en même temps que l'image des objets du décor(n) à l'instant (n+p). On peut en effet calculer une image des objets communs entre les deux que l'on mixe ensuite avec les objets particuliers à chacun.

Il ne reste plus ensuite qu'à réaliser le passage entre le décor(n) et ce qu'il devient à l'instant (n+p). Comme le mixage avec les objets rapides devra être réalisé par la suite, il faut noter qu'il s'agit bien d'une transition entre décors, c'est à dire la combinaison d'une image et d'une carte de profondeur, et non pas d'une simple transition entre deux images.

L'objectif est de rendre ce passage le moins perceptible possible à l'oeil humain au cours de l'animation. Or, comme le remarque Cook dans [COOK86], le système visuel humain est beaucoup moins sensible au bruit qu'à l'aliasage.: s' il est très sensible au déplacement simultané de zones entières de l'image, il semble l'être beaucoup moins à la modification de pixels aléatoirement répartis dans une image qui est modifiée vingt-cinq fois par seconde. De plus, nous savons que, au cours des p images de la période de prédiction, l'apparence des objets du décor a peu changé puisque leur déplacement est inférieur à un pixel: ce sont principalement le contour des objets qui a été modifié comme dans la Figure 46. , page 93. Il suffit en conséquence de répartir les modifications sur plusieurs images pour les rendre moins décelables.

Cette répartition des modifications se passe en deux étapes dont la première est la détection des pixels modifiés au cours de la période de prédiction. Pour cela, on par-

court les images du décor en parallèle et l'on compare les valeurs des couleurs de chaque pixel. Lorsqu'un pixel n'a pas les mêmes valeurs de couleurs pour les deux instants, on ajoute dans un tableau un pointeur sur une structure décrivant son état final (l'instant n+p).

Cette structure doit contenir les informations suivantes:

```
struct {
        short int ligne, colonne;
        unsigned char rouge, vert, bleu;
        double profondeur;
        }
```

Les types des champs dépendent évidemment des caractéristiques des images produites.

A la fin du parcours des images, on dispose donc d'un tableau de taille T permettant d'accéder rapidement à la description des pixels. On peut alors réaliser la seconde phase de la répartition qui est le post-traitement de l'image du décor initial vers celle des mêmes objets à l'instant (n+p).

Entre l'image (n+1) et l'image (n+p), qui est la première image de la période de prédiction suivante, on doit modifier T pixels en p images. Si l'on insère à chaque image la partie entière de (T/p), il peut rester un reliquat important pour la dernière image (ex: si T=247 et p=25 alors on insère à chaque image 9 pixels et il existe un saut de 31 pixels avec la période de prédiction suivante). Afin d'éliminer cet effet, on insérera à chaque image la partie entière du nombre de pixels restants divisé par celui des images venir. En reprenant l'exemple précédent, la suite devient (9 9 9 10 10.... 10). La différence maximale entre deux instants successifs n'est plus alors que de un pixel.

A partir de l'image (n+1), l'insertion des pixels est faite successivement et de manière aléatoire: on tire au hasard un nombre entre 0 et la taille courante du tableau de pixels à insérer moins un, ce qui élit le pixel suivant. On modifie l'image et la carte de profondeur du pixel correspondant en leur affectant les valeurs terminales que l'on a précédemment stockées dans la structure. Afin de retrouver une situation de début d'insertion équivalente à la précédente, on copie le dernier pointeur du tableau à la position de celui qui vient d'être utilisé et on diminue la taille du tableau de un. Ceci jusqu'à épuisement du nombre modification de pixels alloué.

Le cas du temps réel

Dans le cas où les images sont affichées aussitôt produites, le post-traitement décrit ci-dessus ne peut avoir lieu. Puisque l'on ne connaît pas l'avenir avec certitude, il n'y a pas d'autre solution que de faire confiance à la capacité de prédiction du logiciel.

Pour cela, on utilise la même hypothèse que lors du processus de segmentation: les vitesses des objets sont constantes au cours de la période de prédiction. On peut ainsi prédire les positions probables des objets du décor pour l'instant (n+p). A partir de là, on reprend les techniques exposées dans le cas du post-traitement en les appliquant à ce décor probable.

Dans ce cas du traitement à la volée des images, nous ne pouvons en conséquence pas donner d'information sur le saut qui sera effectué à la transition entre les périodes de prédiction et en conséquence sur la qualité de l'animation produite. Dans un très grand nombre de cas cependant, l'hypothèse faite est vérifiée et le traitement a-priori du décor permet de réduire les imperfections de l'animation.

En conclusion, nous avons donc proposé une technique de traitement des décors qui permet la création d'animations non pas identiques mais équivalentes à celles qui seraient produites en considérant les images indépendamment les unes des autres. On entend ici par équivalentes des animations que le système visuel ne peut différencier mais qui peuvent cependant ne pas être constituées des mêmes images à quelques pixels près. Cette équivalence est très bonne dans les applications où un post-traitement est possible mais peut l'être moins dans le cas du temps réel, en fonction de la régularité du mouvement des objets.

III-3-3) Premiers résultats

Nous présentons ici les résultats en terme de temps de calcul de l'application de notre algorithme dans les deux situations les plus courantes dans les films d'animation: le plan fixe et le déplacement de l'observateur au sein d'un décor.

III-3-3-1) Le cas du plan fixe

L'observateur est immobile ainsi que la plupart des objets. Seul un objet (le héros!) se déplace.

Nous prendrons comme exemple un avion qui se rapproche de l'observateur.

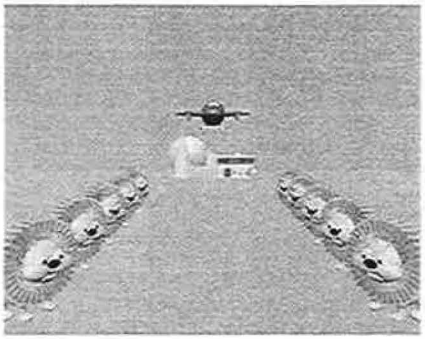
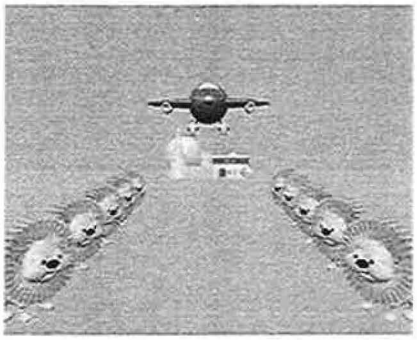
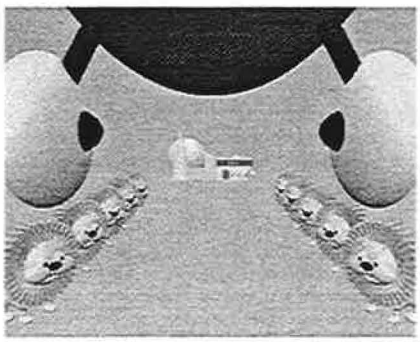




TABLEAU 2. : Comparaison des temps de calcul: observateur fixe, images non ombrés. (Durée de validité = v)

Résolution de l'image	128x128	256x256	512x512	1024x1024
Calcul image par image	9.51	38.16	153	622
Calcul par cohérence	0.47+8/v	3.53+34/v	14.9 + 133/v	63+564/v
Accélération maximale (v infinie)	20	10,8	10,3	9.9

Etant donné que les objets du décor sont immobiles, la durée de vie de celui-ci est infinie. C'est pour cela que les temps de calculs par notre méthode dépendent de la durée de validité v choisie.

III-3-3-2) L'observateur mobile

Cette fois-ci, l'observateur se déplace au sein d'un nuage d'objets identiques uniformément répartis. Nous avons pris comme exemple un nuage de cent coccinelles.

Image type

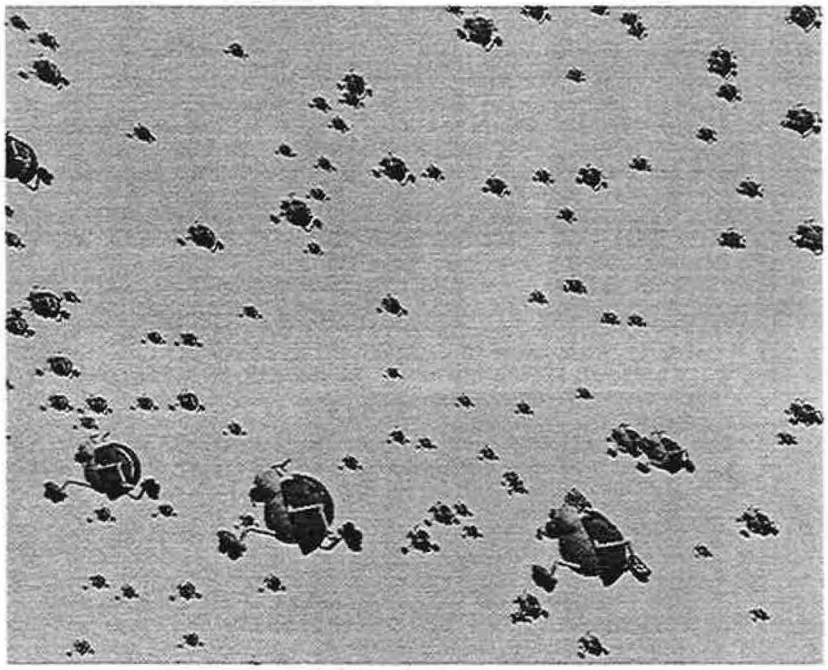


TABLEAU 3. : Comparaison des temps de calcul: observateur mobile, images non ombrées en fonction de la taille des images. (Durée de validité = 25)

Résolution de l'image	128x128	256x256	512x512	1024x1024
Calcul image par image	41.14	163	663	2680
Calcul par cohérence	6,44	73	403	1616
Accélération	6,4	2,2	1,6	1,6

Il est logique de voir l'accélération diminuer quand la résolution de l'image augmente car les mouvements autorisés sont alors beaucoup moins importants.

L'ensemble des algorithmes décrits dans ce chapitre permet cependant de produire des animations de bonne qualité en des temps inférieurs à ceux obtenus avec les

logiciels classiques de création d'animation qui traitent indépendamment les unes des autres une succession d'images.

TABLEAU 4. : Comparaison des temps de calcul: observateur mobile, images non ombrées en fonction de l'erreur permise (Durée de validité = 25, Taille 160x128)

Erreur permise (en pixels)	1	2	4	8
Calcul image par image	51,6	51,6	51,6	51,6
Calcul par cohérence	11,7	6,34	3,3	2,5
Accélération	4,4	8,1	15,6	20,6

De manière symétrique au tableau précédant, l'accélération augmente quand l'on relâche la contrainte sur l'erreur de mouvement autorisée.

Cependant, seuls les rayons primaires sont pour l'instant traités, comme d'ailleurs dans la totalité des logiciels de calcul d'animation temps réel, où le coût de l'ombrage, même sommaire, est trop important. Les ombres portées sont cependant très importantes pour le réalisme des animations produites car elles sont un élément important de notre perception de la réalité en facilitant au système visuel la reconstruction de la profondeur suggérée par la perspective présente dans l'image. La suite de ce chapitre tentera en conséquence d'augmenter le réalisme de l'animation en permettant le calcul d'ombres portées, tout en conservant les autres objectifs fixés dans son introduction: une accélération importante des calculs et la liberté de mouvement des objets et de l'observateur.

III-3-4) Où les ombres rodent

A la suite de la segmentation, ce sont deux bases de données indépendantes qui sont créées. Ainsi, lors de la phase suivante de la création des images, les objets de l'une ne peuvent plus interagir avec ceux de l'autre et, si l'on peut calculer les ombres

portées à l'intérieur de chacune des bases de données, il est impossible de les calculer directement entre celles-ci.

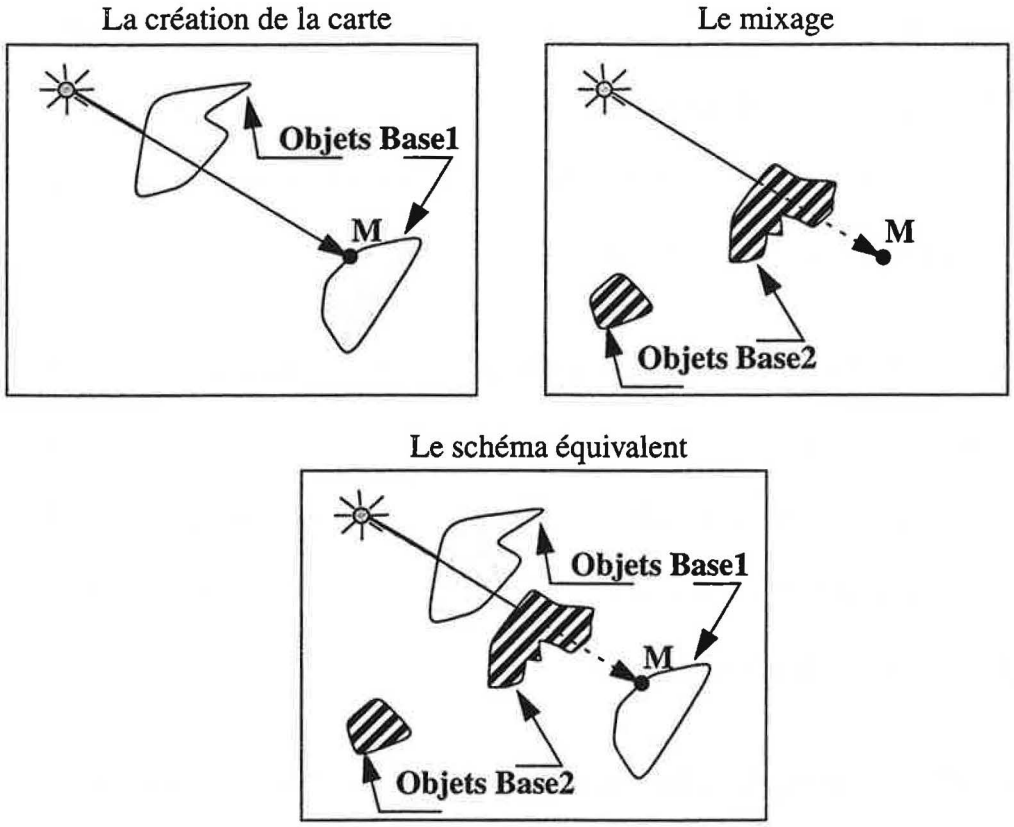
Afin de pouvoir calculer ces ombres entre les bases de données au moment de la phase de mixage, il nous faut en conséquence conserver des informations supplémentaires concernant l'énergie reçue en chaque pixel. Ces informations, par analogie avec les cartes de profondeurs, seront stockées dans ce que l'on nommera par la suite «carte d'ombrage». Le résultat produit à partir de chaque base de données est maintenant la réunion d'une carte de profondeur et d'une carte d'ombrage.

Lors du calcul de l'image correspondant à une base de données et en chaque pixel où il y a une intersection avec la scène, on calcule l'énergie arrivant depuis chaque source ainsi que des informations sur la surface rencontrée (normale, couleur propre, etc...). C'est l'ensemble de ces informations qu'il faut pouvoir retrouver par la suite. La procédure de mixage est en conséquence modifiée de la manière suivante:

- On choisit comme dans le mixage classique grâce à la carte de profondeur quelle base de données va fournir les informations pour le pixel. On la nommera «première base de données».
- Grâce aux informations sur la profondeur et le rayon lancé, on peut retrouver la position réelle (en trois dimensions) de l'élément de la surface intersectée. On crée alors un point matériel fictif qui a les mêmes caractéristiques géométriques que la surface intersectée.
- On plonge finalement ce point fictif dans une scène dont les objets sont ceux de la seconde base de données. Les sources de lumière sont quant à elles remplacées par des sources de même géométrie mais qui n'émettent que l'énergie reçue au travers de la première base de données. C'est à dire que l'on renvoie des

rayons d'ombre vers les sources en tenant compte de l'énergie déjà absorbée au travers de la première base de donnée.

Figure 47. **Le principe de la carte d'ombrage**



Dans la figure ci-dessus, le cadre en haut à gauche correspond à la création d'une base de données. On stocke dans cette dernière l'énergie reçue depuis la source à travers la première base de donnée ainsi que les caractéristiques de la surface en M.

Le cadre en haut à droite représente la phase de mixage: la source lumineuse n'émet que l'énergie qui parvenait en M à travers la première base de donnée et l'on calcule quelle est la part de cette énergie qui parvient en M à travers la seconde.

L'énergie reçue par le pixel est en conséquence celle qui traverse la première base de données puis la seconde. C'est donc bien l'énergie qui traverserait l'union des deux bases de données, comme dans le cadre du bas.

A partir des informations d'énergie arrivante et de géométrie de la surface, le calcul de la couleur du pixel est alors effectué de la même manière que le logiciel de synthèse d'images. Contrairement à précédemment, l'écriture du mixage dépend ainsi de l'algorithme utilisé pour le calcul des images.

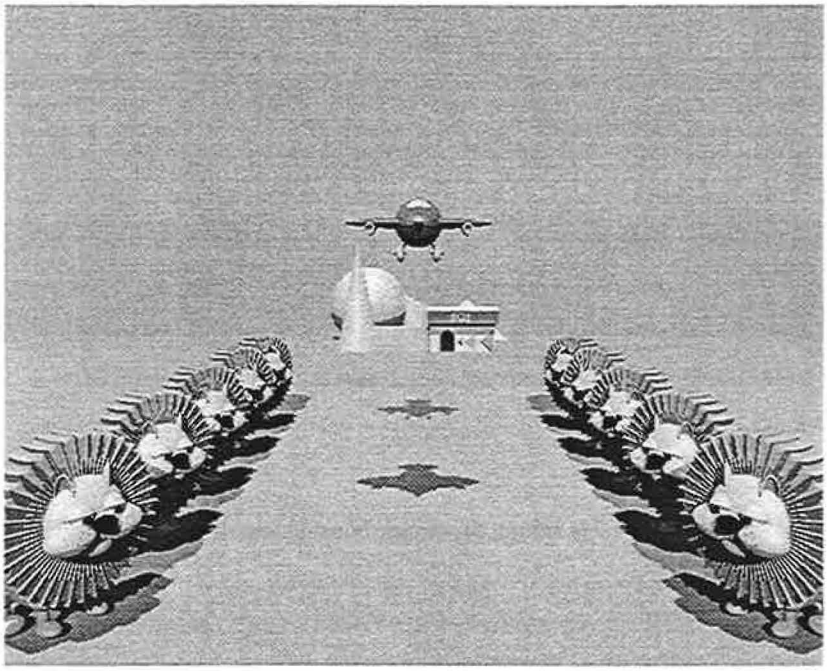
Réciproquement, le mixage exige une modification de ce logiciel de production d'images. En effet, si les cartes de profondeurs sont très souvent produites, ces cartes d'ombrages sont beaucoup plus particulières. Elles doivent fournir, pour chaque pixel où de la matière existe, quelle est l'énergie (pour nous, trois couleurs: rouge, vert et bleu) qui arrive en ce point depuis chaque source.

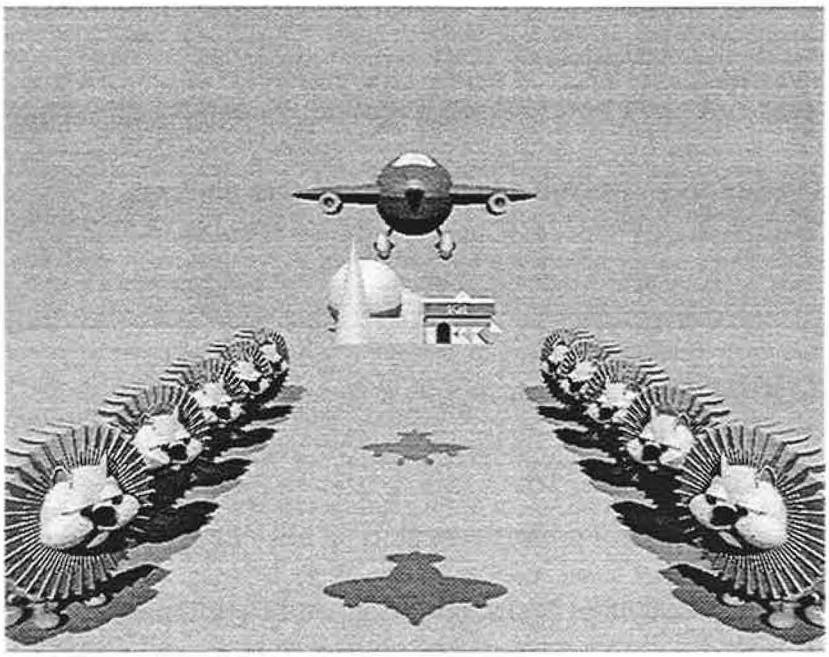
Lors de la phase de mixage, on lancera beaucoup de rayons mais sur peu d'objets depuis la base de données «lente» puisque l'on espère que les objets «lents» sont beaucoup plus nombreux que les objets «rapides». Depuis la base de données «rapide», on lancera peu de rayons mais sur beaucoup d'objets. Les ombrages internes à la base de données «lente» sont quant à eux calculés une fois pour toute. Les gains sur le nombre de rayons d'ombre lancés sont donc comparable à ceux des rayons primaires.

La phase de traitement du décor est quant à elle réalisée de manière identique à la précédente, si ce n'est qu'il faut modifier les cartes d'ombrage en parallèle avec les cartes de profondeurs.

III-3-5) Les résultats du calcul d'animations ombrées

Nous reprenons les bases de données de III-3-3), page 98. L'animation est cette-fois ci calculée avec des ombres.





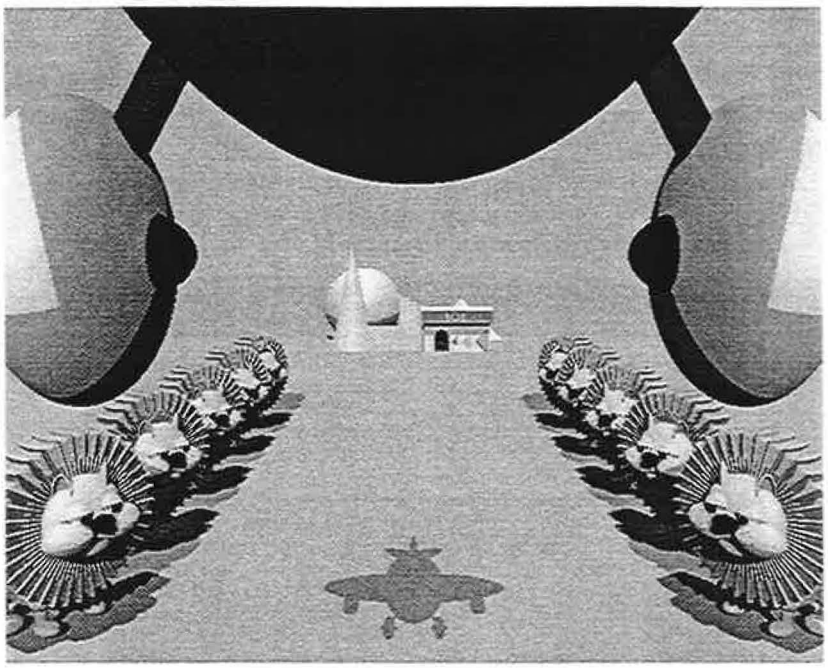


TABLEAU 5. : Comparaison des temps de calcul: observateur fixe, image ombrée

	128x128	256x256	512x512	1024x1024
Calcul image par image	15,8	64.6	262	1049
Calcul par cohérence	1,25 + 21/v	6.4 + 84/v	26,2 + 338/v	104+1354/v
Accélération maximale (validité infinie)	12,6	10,1	10	10

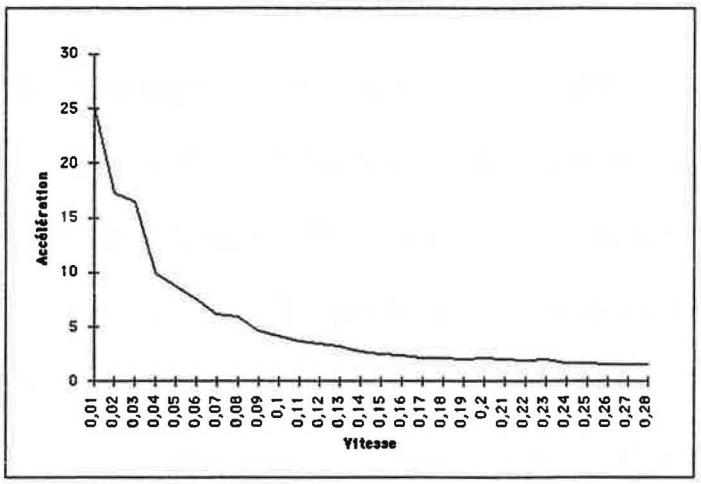
Les résultats sont identiques dans le cas d'un observateur mobile: l'accélération du calcul d'animations ombrées est semblable à celle obtenue sans ombrage.

III-4) Conclusions

Les résultats obtenus semblent prometteurs mais un travail important reste à faire. Il est par exemple important d'étudier le comportement de l'algorithme en fonction de ses divers paramètres.

Evolution de l'accélération en fonction de la vitesse des objets.

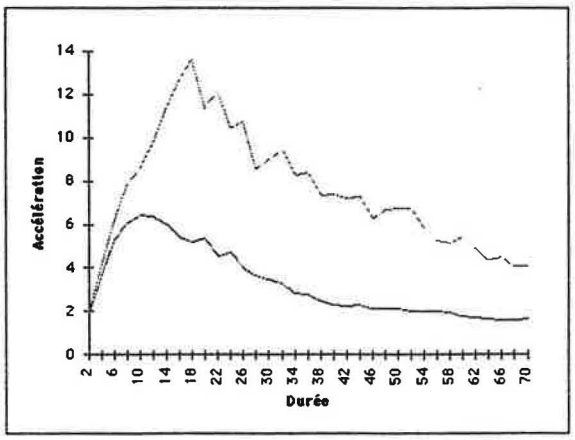
Nous avons étudié ce paramètre en faisant varier la vitesse de déplacement de l'observateur dans le nuage de cent coccinelles. Le résultat est le suivant:



La vitesses est le déplacement de l'observateur entre deux images, les coccinelles ont une taille de 10. Comme on pouvait s'y attendre, l'accélération obtenue diminue quand le mouvement des objets augmente.

Evolution de l'accélération en fonction de la durée de la prédiction.

Les courbes ci-dessous ont été obtenues pour le champ de coccinelles à différentes vitesses:



La forme des courbes est cependant générale. En effet, pour une période courte, on ne tire pas suffisamment partie de la cohérence, ce qui explique une accélération faible. Pour une période longue au contraire, le nombre d'objets dans le décor et avec lui l'efficacité diminuent.

Il serait important de connaître cette courbe a priori afin d'optimiser le choix de la période de validité. Pour une période fixée, il serait rapide de calculer une approximation de l'aire occupée par les objets dans l'écran en remplaçant chaque ellipse par son rectangle englobant et en les accumulant dans l'image. En prenant comme hypothèse que les temps de calcul seront proportionnels à ces aires, une approximation de la courbe peut être tracée. L'implémentation de cette idée est actuellement en cours.

Conclusions

Les premiers résultats obtenus prouvent l'intérêt de notre approche. Son efficacité dépend bien entendu du mouvement présent dans la scène. Elle se montre très intéressante pour les animations classiques en plan fixe avec quelques personnages mobiles. Elle reste cependant intéressante pour la plupart des animations. Elle permet en effet de calculer des animations de synthèse, avec une accélération potentielle importante, sans être limité ni par le mouvement des objets ni par celui de l'observateur. Elle ne nécessite pas de connaître l'animation a priori et peut être adaptée à de nombreux logiciels de synthèse d'images.

Une amélioration du logiciel existant reste cependant à réaliser en y incorporant les réflexions et réfractions lumineuses multiples. Le stockage de l'arbre des rayons permettrait probablement d'étendre cette technique comme nous l'avons déjà fait pour les rayons d'ombre. Une optimisation importante peut enfin être apportée en choisissant une durée de validité adaptée à chaque animation, comme il a été proposé à la fin du chapitre précédant.

Nous n'avons traité de plus dans ce travail que des aspects géométriques de la modification d'une image et plus précisément de la modification de ses pixels pris un par un. Les aspects psycho-visuels n'ont pas été pris en compte. Il nous semble cependant que, par exemple, un objet dont le contraste est faible avec son entourage a une importance moins grande qu'un objet à fort contraste, car son mouvement est plus difficilement détecté. Ces aspects seraient en conséquence également à étudier.

Références

[ADEL95] Stephen J. Adelson. Larry F. Hodges
«Generating Exact Ray Traced Animation Frames by Reprojection»
IEEE Computer Graphics & Applications, Mai 95, pages 43-52

[AMAN84] John Amanatides
«Ray Tracing with Cones»
ACM Computer Graphics Proceedings 84, 18(3), pages129-135

[ARNA87] Bruno Arnaldi, Thierry Priol et Kadi Bouatouch
«A new sapce subdivision method for ray tracing CSG modelled scenes»
The Visual Computer, Vol 3, No. 2, 1987, pages 98-108

[ARVO87] James Arvo, David Kirk
«Fast Ray Tracing by Ray Classification»
ACM Computer Graphics 21(4) pages 55-64

[ATHE83] P.R. Atherton
«A scan-Line hidden surface removal procedure for constructive solid geometry»
ACM Computer Graphics 17(3), pages 73-82

[BADT88] Sig Badt Jr.
«Two algorithms for taking advantages of temporal coherence in Ray tracing»
The Visual Computer, Vol 4, Septembre 88, pages 123-132

[BERG78] Marcel Berger
«Géométrie tome 4: formes quadratiques, quadriques et coniques»
Editions Cedic/Fernand Nathan, 1978

[CARP84] Loren Carpenter
«The A-buffer, an Antialiased Hidden Surface Method»
ACM Computer Graphics Proceedings 84, 18(3), pages 103-108

[CHAP90] J. Chapman, T.W. Calvert J.Dill
«Exploiting Spatial Coherence in Ray Tracing»
Proceedings of Graphics Interface '90
Canadian Information Processing Society Editor
Canada 1990, pages 196-204

[CHAP91] J. Chapman, T.W. Calvert J.Dill
«Spatio-Temporal Coherence in Ray Tracing»
Proceedings of Graphics Interface '91
Canadian Information Processing Society Editor
Canada 1991, pages 101-108

[CLAR76] J.H. Clark
«Hierarchical geometric models for visible surface algorithms»
Communication of the ACM. Vol. 19, No. 10, Octobre 1976, pages 547-554

[COOK84] Robert L. Cook, Thomas Porter, Loran Carpenter
«Distributed Ray Tracing»
ACM Computer Graphics Proceedings 84, 18(3), pages 137-145

[COOK86] Robert L. Cook
«Stochastic Sampling in Computer Graphics»
ACM Transactions on Graphics, Vol5, No. 1, Janvier 1986, pages 51-72

[CROW82] F. C. Crow
«A more Flexible Image Generation Environment»
ACM Computer Graphics Proceedings 82, 16(3), pages 9-18

[DUFF85] Tom Duff
«Compositing 3-D Rendered Images»
ACM Computer Graphics Proceedings 85, pages 41-44

[FINC75] Christopher Finch
«The Art of Walt Disney»
Walt Disney Prroduction, 1975

[FURH95a] Borko Furht
«A survey of Multimedia Compression and Standards. Part I: JPEG Standard»
Real-Time Imaging, Vol. 1, No.1 1995, pages 49-67

[FURH95b] Borko Furht
«A survey of Multimedia Compression and Standards. Part II: Video Compression»
Real-Time Imaging, Vol. 1, No.1 1995, pages 319,337

[GLAS84] Andrew S. Glassner
«Space Subdivision for Fast Ray Tracing»
IEEE Computer Graphics & Applications, Octobre 84, pages 15-22

[GLAS88] Andrew S. Glassner
«Spacetime Ray Tracing for Animation»
IEEE Computer Graphics & Applications, Mars 88, pages 60-70

[GORT96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, Michael F. Cohen
«The Lumigraph»
ACM Computer Graphics Proceedings 96, pages 43-54

[GREE89] S.A. Green and D.J. Paddon
«Exploiting Coherence for Multiprocessor Ray Tracing»
IEEE Computer Graphics & Applications, Novembre 89, pages 12-26

[GROL91] Eduard Gröller, Werner Purgathofer
«Using temporal and spatial coherence for accelerating the calculation of animation sequences»
Proceeding of Eurographics '91 pages 103-113

[GROL94] Eduard Gröller, Werner Purgathofer
«Coherence in Computer Graphics»
Technical Report, 1994
Institute for Computer Graphics, Technical University Vienna
ftp://ftp.cg.tuwien.ac.at/pub/TR/94/TR-186-2-94-10.Paper.ps.gz

[HECK84] Paul Heckbert and Pat Hanrahan
«Beam Tracing Polygonal Objects»
ACM Computer Graphics Proceedings 84, 18(3), pages 119-127

[HAIN86] Eric A. Haines and Donald P. Greenberg
«The Light Buffer: A Shadow-Testing Accelerator»
IEEE Computer Graphics & Applications, Septembre 86, pages 6-16

[HECK84] Paul S. Heckbert, Pat Hanrahan
«Beam Tracing Polygonal Objects»
ACM Computer Graphics Proceedings 84, 18(3), pages 119-127

[HUMB94] Pascal Humbert, Yvon Gardan
«Utilisation des arbres de rayons pour le rendu de séquence d'animation»
AFIG'94, Toulouse, pages 261-272

[KAY86] Kay and Kajiya
«Ray tracing complex scenes»
ACM Computer Graphics 20(4) pages 269-278, 1986

[JANS83] F.W. Jansen, J.J. van Wijk
«Fast Previewing Techniques in Raster Graphics»
Proceeding of Eurographics '83, pages 195-202

[JOY86] Kenneth I. Joy, Murthy N. Bhetanabhotla
«Ray Tracing Parametric Patches Utilizing Numerical Technique and Ray Coherence»
ACM Computer Graphics 20(4) pages 179-284, 1986

[LEVO77] Marc Levoy
«A color Animation System based on the Multiplane Technique»
ACM Computer Graphics Proceedings 1977, pages 65-71

[GORT96] Marc Levoy and Pat Hanrahan
«Light Field Rendering»
ACM Computer Graphics Proceedings 96, pages 31-42

[LIOU91] M. Liou
«Overview of the P X 64 Kbits/s video coding standard»
Communication of the ACM, 34(4), pages 59-63

[MAIL96] Jean-Luc Maillot
«Pseudo-réalisme et progressivité pour le tracé de rayons»
Thèse, Septembre 1996, Ecole des Mines de Saint-Etienne

[MARK90] J. Marks, R. Christensen and M. Friedell
«Image and Intervisibility Coherence in Rendering»
Proceedings of Graphics Interface '90
Canadian Information Processing Society Editor
Canada 1990, pages 17-30

[MAUR93] Hervé Maurel, Yves Duthen, René Caubet
«A 4D Ray Tracing»
Proceeding of Eurographics '93, pages 285-294

[MURA90] Koichi Murakami, Katsuhiko Hirota
«Incremental Ray Tracing»
Eurographics Seminars
Tutorials and Perspectives in Computer Graphics: «Photorealism in Computer Graphics», Springer Verlag.

[NIME95] Jeffry Nimeroff, Julie Dorsey and Holly Rushmeier
«A Framework for Global Illumination in Animated environments»
6th Eurographics Workshop on Rendering, Dublin, June 95, pages 223-235

[NIME96] Jeffry Nimeroff
«A Temporal Image-Based Approach to Motion Reconstruction for Globally Illuminated Animated Environments»»
7th Eurographics Workshop on Rendering, Porto, 96, pages 175-184

[OHTA90] Masataka Ohta and Mamoru Maekawa
«Ray-bound tracing for perfect and efficient anti-aliasing»
The Visual Computer, Vol 6, 1990, pages 125-133

[PEAR91] A. Pearce, D. Jevans
«Exploiting Shadow Coherence in Ray Tracing»
Proceedings of Graphics Interface '90
Canadian Information Processing Society Editor
Canada 1990, pages 109-116

[PLAN90] H. Plantiga, R. Ch. and B.W. Seales
«Real-Time Hidden-Line Elimination for a Rotating Polyhedral Scene Using the Aspect Representation»
Proceedings of Graphics Interface '90
Canadian Information Processing Society Editor
Canada 1990, page 9-16

[REGA94] Matthew Regan, Ronald Pose
«Priority Rendering with a Virtual Reality Address Recalculation Pipeline»
ACM Computer Graphics Proceedings 1994, pages 155-162

[RUBI80] Steve Rubin and Turner Whitted
«A Three-Dimensional Representation for Fast Rendering of Complex Scenes»
ACM Computer Graphics 14(3), Juillet 1980, pages 110-116

[SEGA92] Mark Segal, Carl Korobkin, Rolf van Widenfelt, Jim Foran, Paul Haeberli
«Fast Shadows and Lighting Effects Using Texture Mapping»
ACM Computer Graphics Proceedings 1992, 26(2) pages 249-252

[SEQU89] Carlo H. Séquin and Eliot K. Smyrl
«Parameterized Ray Tracing»
ACM Computer Graphics 23(3) pages 307-314

[SHIN87] Mikio Shinya, Tokiichiro Takahashi, Seiichiro Naito
«Principles ans Applications of Pencil Tracing»
ACM Computer Graphics Proceedings 87, 21(4), pages 45-54

[SPEE85] R. Speer, T. DeRose, B. Barsky
«A theoretical and Empirical Analysis of Coherent Ray-Tracing»
Proceedings of Graphics Interface '85, pages 11-25

[TOST91] Danièle Tost
«An algorithm of hidden surface removal based on frame-to-frame coherence»
Proceeding of Eurographics '91, pages 261-273

[WALL81] Bruce A. Wallace
«Merging and Transformation of Raster Images for Cartoon Animation»
ACM Computer Graphics Proceedings 81, 15(3), pages 253-262

[WATK70] G.S. Watkins
«A real time visible surface algorithm»
University og Utah UTEC-CSC, 1977, pages 70-101

[WEGH84] Hank Wehgorst, Gary Hooper, and Donald P.Greenberg
«Improves Computational Methods for Ray Tracing»
ACM Transactions on Graphics, Vol. 3, No. 1, January 84, pages 52-69

[WHIT80] Y. Whitted
«An improved illumination model for shaded display»
Communication of the ACM, Vol. 23, No. 6, Juin 1980, pages 343-349

Annexe A

Les formats des fichiers de données et leur exploitation

A-1) Les fichiers d'animation

Les fichiers d'animation décrivent les diverses scènes à visualiser au cours du temps quand celles-ci ne sont pas fournies en temps réel. Il nous ont permis de tester le logiciel. Ce sont des fichiers ASCII dont chaque scène est constituée de deux parties: la description de l'environnement et de celle des objets.

L'environnement est constitué de la caméra, de sa direction de visée, des sources de lumières et des types de rendu (ombré ou non, la taille de l'image, etc...). Les lignes d'environnement débutent par * suivi d'un mot clef et des paramètres correspondants.

*eye x y z vx vy vz	pour l'oeil.
*aim x y z vx vy vz	pour le point de visée.
*angles A B	pour les angles d'ouverture du cône de visée.
*ombres b	Y a-t-il ombrage? (0=non 1=oui)
*taille largeur hauteur	La taille des images produites

*source x y z r v b type + paramètres selon le type.

où x, y et z sont des positions; vx, vy et vz des vitesses; r, v et b les couleurs de la source; type son type (soleil, ponctuelle, etc...);b un booléen (zéro ou un).

Les lignes d'objets débutent par ! suivi de la localisation de la description de l'objet, de sa position en x, y et z ainsi que les trois rotations possibles puis des vitesse correspondantes:

```
!/usr/people/nicolas/BDD/avion x y z a b c vx vy vz va vb vc
```

Les différentes scènes sont séparées par une ligne débutant par @ suivi du numéro de scène.

A-2) Les fichiers de scène

Les fichiers de scène sont la traduction de chaque scène du fichier d'animation fournie au logiciel de synthèse d'images. Ils doivent donc s'adapter au format que nécessite celui-ci. Puisque nous avons utilisé l'environnement Illumines de l'Ecole des Mines, ces fichiers sont des fichiers ASCII à philosophie CSG.

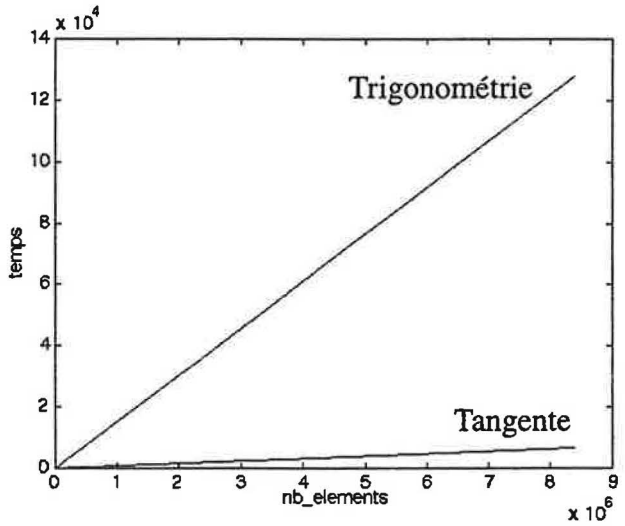
La description des objets est recopiée depuis leur localisation donnée par le fichier d'animation puis on effectue les translations et rotations nécessaires avant de réaliser l'union avec les autres objets de la scène. La philosophie CSG est en conséquence naturelle pour cette application.

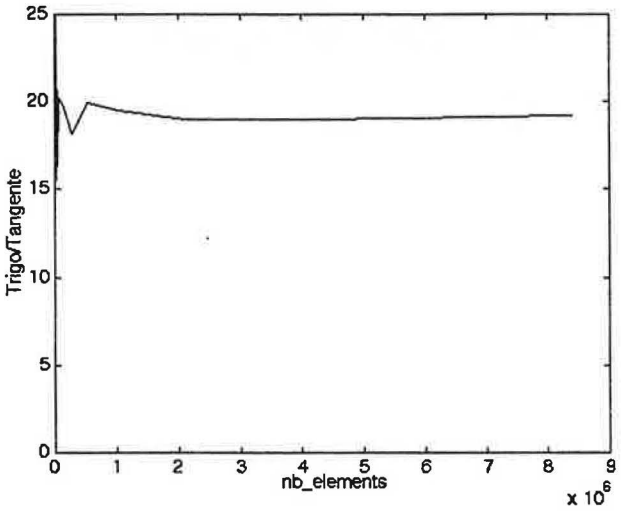
Afin de ne pas multiplier les descriptions des objets, les noms de celles-ci sont stockés dans une table de hachage. Ainsi, si plusieurs objets sont identiques à des transformations près, la phase de traduction le détecte et la description du premier sert aux autres, auxquels on applique seulement les transformations nécessaires.

Annexe B

Implémentation et gain de temps

Voici les temps de calcul du clipping par les méthodes trigonométriques et par les tangentes ainsi que le rapport des temps.





Ces courbes montrent bien comment la même idée, implantée avec des approches différentes, peut engendrer des résultats très diverses. Nous nous sommes appliqués, dans l'ensemble de notre logiciel, à chercher l'implémentation la moins coûteuse en temps de calculs.

Accélération du calcul d'animations de synthèse.

Résumé

Le lancé de rayons est une technique très utilisée pour la production d'animations de synthèse. Les temps de calculs nécessaires sont cependant très importants.Nous proposons en conséquence une méthode permettant l'accélération de ceux-ci.

Nous construirons tout d'abord les outils permettant de mesurer l'influence des translations et rotations de chaque objet sur son mouvement apparent dans l'espace de l'image, ceci entre deux instants de l'animation.

A l'aide de cette information, nous proposerons une méthode permettant le calcul d'une animation visuellement équivalente à celle qui serait produite avec un calcul image par image, en un temps beaucoup plus court. Contrairement à d'autre méthodes, aucune limitation n'existe sur les mouvements ni de l'observateur ni des objets de la scène. De plus, la connaissance a priori de l'animation n'est pas nécessaire, ce qui permet d'appliquer la méthode au temps réel.

Mots clefs

Synthèse d'images, lancé de rayons, animation, cohérence, cohérence temporelle, accélération

Computing animations: A temporal coherence based speed-up.

Abstract

Ray tracing is a well-known technique for generating realistic images. One of the major drawbacks of this approach is the extensive computational requirement for image calculation. We suggest an algorithm to speed this calculation up.

We first build tools to measure, between two instants, the influence of the 3D motion of objects on their 2D image.

We then use this information to calculate a realistic animation whith a good speed-up. There isn't any limitation either on objects or on eye movements. The animation needs not being known at start, and this method may be used in real-time.

Keywords

Computer graphics, ray tracing, animation, coherence, temporal coherence, speed-up, acceleration.