Deux problèmes en synthèse d'images : les sources directionnelles de lumière et une interface évoluée

Gilles Fertey

ECOLE NATIONALE SUPERIEURE
DES MINES DE SAINT-ETIENNE

UNIVERSITE DE SAINT-ETIENNE
N° d'ordre : 40 ID

THESE

présentée par

Gilles FERTEY

pour obtenir le titre de

DOCTEUR

DE L'UNIVERSITE DE SAINT-ETIENNE

ET DE L'ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE

(Spécialité : Informatique, Image, Intelligence Artificielle et Algorithmique)

Deux problèmes en synthèse d'images :

Les sources directionnelles de lumière et Une interface évoluée

Soutenue à Saint-Etienne le 30 Mai 1990

COMPOSITION DU JURY

Monsieur Denis VANDORPE		Rapporteurs
Monsieur Michel LUCAS		
Madame	Eunika MERCIER-LAURENT	Examinateurs
Messieurs	Jean AZEMA	
	Christian BERTIN	
	Bernard PEROCHE	

ECOLE NATIONALE SUPERIEURE
DES MINES DE SAINT-ETIENNE

UNIVERSITE DE SAINT-ETIENNE
N° d'ordre : 40 ID

THESE

présentée par

Gilles FERTEY

pour obtenir le titre de

DOCTEUR

DE L'UNIVERSITE DE SAINT-ETIENNE

ET DE L'ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE

(Spécialité : Informatique, Image, Intelligence Artificielle et Algorithmique)

Deux problèmes en synthèse d'images :

Les sources directionnelles de lumière et Une interface évoluée

Soutenue à Saint-Etienne le 30 Mai 1990

COMPOSITION DU JURY

Monsieur Denis VANDORPE		Rapporteurs
Monsieur Michel LUCAS		
Madame	Eunika MERCIER-LAURENT	Examinateurs
Messieurs	Jean AZEMA	
	Christian BERTIN	
	Bernard PEROCHE	

Remerciements

Je tiens à exprimer ma reconnaissance aux personnes qui ont accepté de juger ce travail ainsi qu'à toutes celles qui m'ont aidé au cours de ces années de thèse.

- Monsieur le Professeur Bernard Péroche, Directeur du Département Informatique Appliquée de l'Ecole des Mines de Saint-Etienne, de m'avoir accueilli au sein de son équipe ainsi que pour sa compétence, sa disponibilité et la formation que j'ai reçue au cours de ces années de thèse. Ses remarques, son apport ont été sources d'approfondissement pour l'étude que nous avons menée.

- Messieurs les Professeurs

  Michel Lucas, de l'Ecole Nationale Supérieure de Mécanique de Nantes
  et Denis Vandorpe, de l'Université Claude Bernard de Lyon

  pour avoir accepté la lourde charge d'être rapporteurs de cette thèse. Leurs nombreuses remarques tant au niveau de la forme que du contenu m'ont permis d'améliorer la première version du manuscrit.

- Madame Eunika Mercier-Laurent, Ingénieur au sein de la filière Enseignement Supérieur et Recherche de la compagnie Bull, pour sa participation à ce jury.

- Monsieur Jean Azéma, Maître de Conférence à l'Université de Saint-Etienne qui a bien voulu se joindre au jury.

- Monsieur Christian Bertin, Ingénieur à l'Ecole des Mines de Saint-Etienne, pour sa compétence, sa rigueur et l'aide qu'il m'a apportée, notamment en intelligence artificielle.

- Tous les membres passés, présents de l'équipe Images, pour leurs supports moral, scientifique et linguistique (n'est-ce pas Jean-Michel ?). Que Jacqueline Argence-Tallot, Michel Beigbeder, Helmi Ben Amara, Annie Corbel-Bourgeat, Djamchid Ghazanfarpour, Gabriel Hanoteaux, Ghassan Jahmi, Roland Jegou, Dominique Michelucci, Jean-Michel Moreau, Zhigang Nié, Philippe Jaillon, Mohand Ourabah Benouamer, Marc Roelens et Pascale Roudier trouvent ici mes plus vifs remerciements.

- Monsieur Philippe Vial pour ses conseils et son aide pendant l'utilisation de KOOL.

- Tous les membres du Département Informatique Appliquée pour leur amitié et leur aide. Plus particulièrement Jean Pierre Tavernier pour la maintenance du matériel informatique et quelques sauvegardes qui furent les bienvenues; Marie-Line Barnéoud pour sa disponibilité et son assistance typographique.

- Le personnel du service de reprographie pour avoir assuré à plusieurs reprises la réalisation de cet ouvrage.

- Ma mère, mon père, mon frère, les parents et les amis pour leur affection et leur soutien.

Chapitre 3 : Une interface évoluée de CASTOR

INTRODUCTION

La réalisation d'une image de synthèse s'effectue en deux étapes. La première consiste à introduire dans la mémoire de l'ordinateur les informations nécessaires à la définition de la scène. Le concepteur de l'image crée un "modèle" de la scène, ceci à partir de différents outils mis à sa disposition. La modélisation de la scène (ainsi nomme-t-on cette première étape) constitue pour le concepteur un travail important. La deuxième étape consiste à exploiter les données introduites lors de la modélisation, afin de produire une "image" de la scène. Les algorithmes d'élimination des parties cachées et de rendu représentent la partie "visualisation", seconde étape nécessaire à la réalisation de notre image, durant laquelle le concepteur n'intervient alors plus (ou presque). Ces deux étapes forment véritablement deux entités distinctes qui ont fait l'objet, parallèlement, de beaucoup d'études. Des résultats importants ([PERO 88]) ont été acquis dans les deux domaines, dans de plus grandes proportions il est vrai, en ce qui concerne la visualisation. La recherche d'un plus grand réalisme a en effet entraîné diverses études très poussées notamment au niveau du calcul de l'éclairement. En effet, si nous voyons un objet dans la réalité, c'est parce qu'il est éclairé. La lumière joue donc un rôle primordial. Elle est présente sous diverses formes et évolue dans l'environnement d'une façon différente selon les conditions de l'atmosphère, de la nature des objets et bien sûr, des caractéristiques des sources de lumière. Afin de simuler la réalité, de nombreux travaux en synthèse d'images sur la création de modèles d'éclairement et sur la modélisation de sources de lumière, ont été réalisés.

En visualisation comme en modélisation, on assiste de nos jours à l'utilisation d'autres techniques.

Le besoin de machines de plus en plus rapides pour la visualisation a introduit ces dernières années une recrudescence de la recherche vers des méthodes utilisant le parallélisme.

La difficulté et la lourdeur de la modélisation montre la nécessité d'une aide plus importante de la machine. L'intelligence artificielle semble être un outil intéressant, de nombreux travaux sont en cours dans ce domaine.

Les recherches à l'EMSE portent sur l'ensemble de la synthèse d'image, modélisation comme visualisation. Notre système interne de synthèse tridimensionnelle, ILLUMINES ([BEIG 90]), est centré sur un modeleur du type arbre de construction : **CASTOR** ([BEIG 86], [BEIG 88]). Plusieurs interfaces, plusieurs techniques de visualisation peuvent être utilisées (des schémas récapitulatifs seront présentés tout au long de cette thèse, sur la partie saisie et la partie visualisation d'ILLUMINES). Le concepteur

d'image peut ainsi visualiser rapidement, en cas d'interaction, ou au contraire produire une image présentant un certain réalisme (ombres, transparences, sources de lumière, ...) avec un temps de calcul, bien sûr, plus important.

Dans le premier chapitre, nous détaillerons les modèles d'éclairement existants. Nous essayerons de décrire au mieux les modèles basés sur des lois de la physique. Nous proposerons également de classer les modèles d'éclairement, suivant leurs caractéristiques et la manière dont ils prennent en compte la lumière.

Le chapitre deux sera consacré aux modèles de sources de lumière. Nous insisterons, au début de ce chapitre, sur la description physique des sources lumineuses ainsi que leur prise en compte en synthèse d'images. Nous présenterons les modèles existants puis nous développerons notre modèle.

Le chapitre trois est l'étude et la conception d'une interface évoluée pour la création d'images de synthèse. Nous présenterons les objectifs ainsi que les spécifications de l'interface souhaitée. Nous analyserons les différents travaux existants, en synthèse d'images ou dans les domaines sensiblement voisins que sont la Robotique, la géométrie bidimensionnelle et la CAO/CFAO. Nous décrirons enfin de façon plus précise la réalisation de cette interface évoluée : son état actuel et ses extensions.

Chapitre 1

Les modèles d'éclairement

1. Introduction

Le but de la synthèse d'images réalistes est de générer des images qui provoquent des sensations visuelles identiques à celles provoquées par une scène réelle. A ce titre, le calcul de l'éclairement de la scène visualisée est primordial. Si nous voyons un objet rouge, transparent et ayant la forme d'une sphère et non pas d'un disque, c'est dû à l'aptitude de l'objet à renvoyer la lumière d'une certaine façon. Dans la réalité, la lumière, les couleurs que nous percevons, sont des phénomènes physiques régis par des lois bien précises. Aussi, le calcul de l'éclairement en synthèse d'images est basé sur des modèles décrivant de manière plus ou moins précise cette réalité.

Dans le paragraphe 2 nous rappellerons la nature de la lumière et comment elle est interprétée en synthèse d'images.

Le paragraphe suivant sera consacré à une présentation détaillée et à une classification des divers modèles d'éclairement introduits en synthèse d'images. Afin de bien dissocier le problème du rendu des couleurs de celui du calcul de l'éclairement, nous ne parlerons durant ce paragraphe que d'énergie et non pas de couleur.

Le principe d'un modèle d'éclairement consiste à simuler les mécanismes physiques qui régissent la propagation de la lumière et son interaction avec les objets composant une scène.

Ces mécanismes sont connus mais très complexes. Afin d'obtenir une image fidèle de la réalité avec des temps de calcul acceptables, certaines simplifications s'imposent.

La simulation de la propagation de la lumière à travers l'environnement reste, même avec simplification, un problème complexe. L'énergie incidente de la lumière arrivant sur une surface de la scène est décrite comme étant la somme de la lumière directement émise par les sources lumineuses, et de la lumière réfléchie par les autres surfaces de l'environnement. Cette énergie indirecte porte le nom d'illumination globale ; elle fut longtemps remplacée par l'utilisation d'une constante uniforme pour toute la scène.

Les premiers modèles d'éclairement utilisés étaient purement locaux : la lumière émise en un point ne dépendait que de paramètres locaux (réflectance, direction des lampes, ...) et était indépendante de la lumière émise par tous les autres points. Ces modèles sont très proche de la réalité dans le cas d'un objet éclairé et isolé dans un coin de l'espace. Nous détaillerons les plus importants d'entre eux dans le paragraphe 3.1.

Les modèles permettant de tenir compte de la lumière de toutes les surfaces prennent le nom de modèles globaux. Par la détermination de l'intensité d'une lumière quittant une surface dans toutes les directions, le calcul de l'illumination globale est possible. Nous parlerons dans le paragraphe 3.2 des premières méthodes utilisées et de leurs limites. Pour conclure ce paragraphe, nous citerons les tendances actuelles de la recherche.

2 la lumière en synthèse d'images

2.1 Quelques notions sur la lumière

La lumière provenant d'une source lumineuse est due à un phénomène physique. Les sources lumineuses transforment une énergie (calorifique, électrique par exemple) en rayonnements électromagnétiques. La lumière est donc une énergie électromagnétique.

Le *flux énergétique* ϕ_e de la lumière, ie l'énergie rayonnée par unité de temps qui traverse une surface éclairée, est déterminé par la ou les longueurs d'onde de la distribution spectrale de ladite lumière. A ce flux énergétique est associé le *flux lumineux* qui tient compte de la sensibilité relative $V(\lambda)$ de l'œil normal moyen humain :

• dans le cas d'une source monochromatique (la distribution est constituée d'une seule longueur d'onde λ_0), le flux lumineux est défini par:

$$\phi = 680\ V(\lambda_0)\ \phi_e(\lambda_0)$$

• dans le cas d'une source à spectre continu, le flux lumineux est

$$\phi = 680 \int_0^\infty V(\lambda)\ \phi_e(\lambda) d\lambda$$

où $\phi_e(\lambda)$ est le flux énergétique monochromatique correspondant aux radiations comprises entre λ et $\lambda + d\lambda$.

A partir du flux lumineux, plusieurs grandeurs caractéristiques de la photométrie peuvent être définies. Ce sont, entre autres :

- l'*intensité* I, qui est la quantité de flux lumineux par unité d'angle solide ;

- l'*éclairement* E, qui est la quantité de flux lumineux par unité de surface. Par un jeu simple d'équations, l'éclairement peut s'exprimer en fonction de l'intensité :

$$E = I\frac{\cos\theta}{r^2}$$

où

θ est l'angle formé par la droite joignant la source et le centre de l'élément de surface et le vecteur normal à l'élément de surface.

Pour $\theta = 0$, la surface est éclairée au maximum, pour $\theta = \frac{\pi}{2}$, l'éclairement est rasant et donc nul.

r est la distance de la surface à la source.

- la *luminance* , qui est la quantité d'intensité par unité de surface apparente. Elle dépend de l'angle entre la direction de l'observateur et la direction normale à la surface.

La *réflectance* est le rapport de l'énergie de l'onde réfléchie sur celle de l'onde incidente.

La *transmittance* est le rapport de l'énergie de l'onde transmise sur l'énergie de l'onde incidente.

2.2 Ce que nous voyons

L'œil humain, ou plutôt le système visuel humain (œil, nerf optique, cerveau) est sensible à la lumière. Ce que nous voyons est la lumière arrivant en tous les points de la rétine. Cette sensation se traduit par trois réponses de l'œil :

- une réponse à la qualité de la lumière : la sensibilité de l'œil à la luminosité a été démontrée expérimentalement et résulte de l'ouverture de l'iris et de la sensibilité de la rétine. Elle a pour nom : la clarté.

- une réponse à la longueur d'onde : dans une bande située entre 380 et 765 nanomètres, le spectre visible, tout rayonnement électromagnétique est traduit par notre système en sensations lumineuses. L'énergie relative de chaque longueur d'onde possède une apparence caractéristique de couleur : par exemple autour de 400 nanomètres, on distingue du bleu. La perception de la couleur est alors obtenue par la somme de chacune des sensations relatives à chaque longueur d'onde. Cette propriété de l'œil est considérée comme la plus importante.

- une réponse à la structure spectrale de l'image : cette propriété, plus complexe, a permis de constater que la perception de couleur est purement psycho-physiologique. Suivant les conditions de vision (contraste, distance de l'œil à l'image,...), deux couleurs de même longueur d'onde ne provoquent pas forcément la même sensation pour le système visuel humain. Cette interprétation psychosensorielle porte un nom, l'acuité visuelle, et est propre à la vision humaine.

2.3 En synthèse d'images

Ce que nous voyons est la lumière, une distribution spectrale, vue en tous les points de la rétine. Par analogie, la génération d'images de synthèse doit calculer la distribution spectrale d'énergie en chaque point de la scène. Les matériels informatiques (écrans de couleur) possèdent un système trichromatique de reproduction des couleurs. Une conversion de la distribution spectrale, une distribution continue, dans un système trichromatique est nécessaire.

Pratiquement on calcule la distribution spectrale de l'énergie sur un ensemble fini de longueurs d'onde. L'échantillonnage des valeurs choisies peut varier d'où un nombre plus ou moins important de valeurs. Une perte d'information sera inévitable. Le paragraphe 2.3.2 insistera sur ce problème.

2.3.1 La théorie trichromatique des couleurs

Il a été démontré expérimentalement que toute couleur peut être représentée par la combinaison de trois couleurs de base, ou primaires. Afin de pouvoir reproduire le plus grand nombre possible de couleurs, ces trois primaires doivent vérifier les propriétés suivantes :

- une couleur primaire ne peut être obtenue à partir d'une combinaison des deux autres,

- la représentation d'une couleur par les trois couleurs de base doit être unique.

Basés sur cette théorie trichromatique, plusieurs systèmes, ou modèles, de représentation des couleurs ont été créés.

2.3.1.1 Le système RVB de la CIE

Les éléments photosensibles de l'œil sont constitués de deux sortes d'éléments:

- les bâtonnets, très sensibles, qui n'interprètent que la lumière et, de ce fait, interviennent lors de la vision nocturne.

- les cônes, peu sensibles, qui sont responsables de la vision diurne. Ils se répartissent en trois types, chacun d'eux ayant un spectre d'absorbtion dans le rouge, le vert et le bleu. Ceci permet d'expliquer la sensibilité de l'œil aux couleurs.

Par analogie, la Commission Internationale de l'Eclairage a défini un repère de couleurs dont les composantes de base sont le Rouge, le Vert et le Bleu, avec comme longueurs d'onde respectives :

Rouge 700 nanomètres

Vert 546 nanomètres

Bleu 435.8 nanomètres

Ce modèle RVB utilise un système de coordonnées cartésiennes qui décrit un espace de couleurs contenu dans un cube (voir la figure 1).

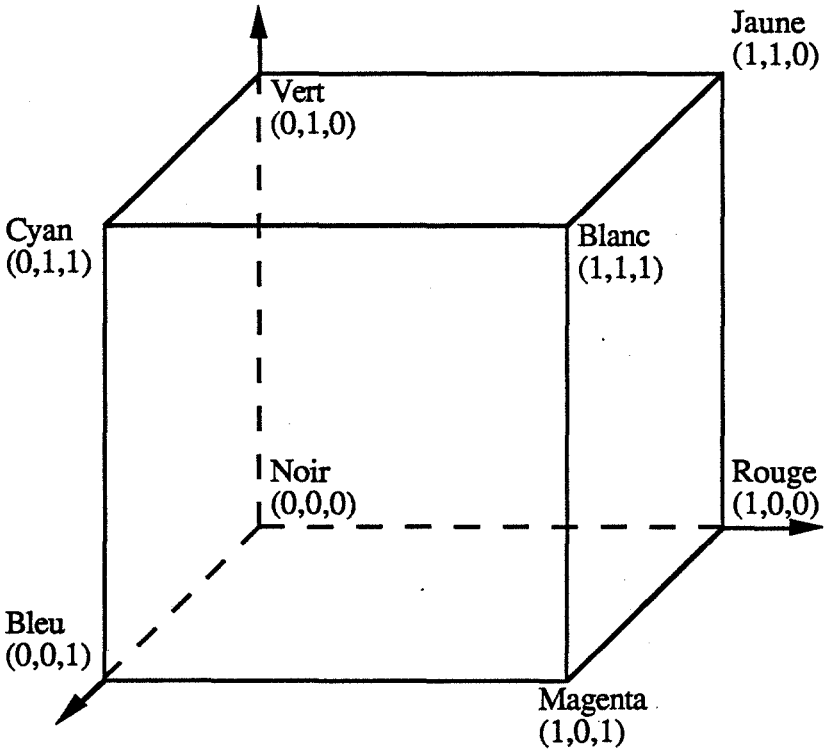


Figure 1

Ces trois couleurs de base, utilisées en télévision, permettent de reproduire un nombre important de couleurs.

2.3.1.2 Le système XYZ de la CIE

La CIE a également normalisé un autre système dont les composantes primaires sont indépendantes du moniteur. Ces trois composantes X Y Z sont respectivement obtenues par intégration du produit de la distribution spectrale d'énergie par des coefficients de distribution $\bar{x}(\lambda)$, $\bar{y}(\lambda)$ et $\bar{z}(\lambda)$ dont la forme est décrite par la figure 2. Ces coefficients possèdent les caractéristiques suivantes :

- $\bar{y}$ (λ) n'est autre que la sensibilité d'un observateur moyen de référence. La courbe $\bar{y}(\lambda)$ a donc la même allure que la courbe expérimentale de réponse de l'œil à la luminosité ;

- les trois courbes n'ont que des valeurs positives.

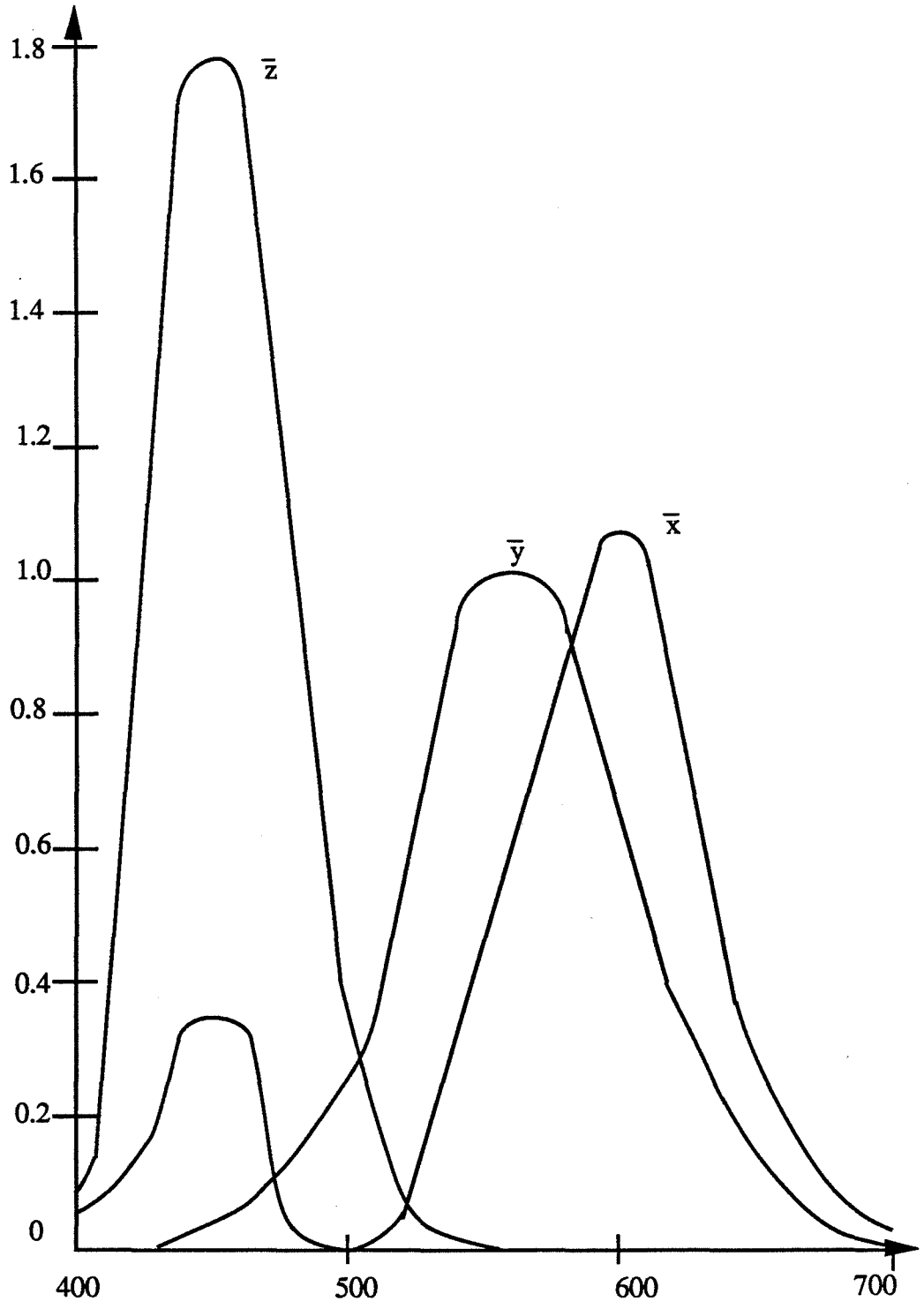


Figure 2 : les coefficients de distribution du système XYZ

Le repère XYZ étant indépendant du moniteur utilisé, une conversion des couleurs dans le système RVB du moniteur est nécessaire. Cette transformation n'est autre qu'un changement de repère. Elle s'effectue par un produit de matrice :

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} X_r & X_v & X_b \\ Y_r & Y_v & Y_b \\ Z_r & Z_v & Z_b \end{pmatrix} \begin{pmatrix} R \\ V \\ B \end{pmatrix}$$

où R, V et B sont les coordonnées de la couleur dans le repère RVB, comprises de 0 à 1. X,Y et Z représentent les coordonnées de cette couleur dans le repère XYZ. Les éléments Xr, Xv, Xb,Yr, Yv, Yb, Zr ,Zv et Zb sont déterminés à partir des coordonnées chromatiques des luminophores du moniteur.

2.3.1.3 Le système TSL

La réponse de l'œil aux longueurs d'onde a été considérée jusqu'à présent comme primordiale pour la définition des systèmes précédents. D'autres modèles de couleurs sont, au contraire, basés sur l'expérience intuitive de la perception de la couleur. Le modèle TSL ([SMIT 78]) en est un exemple. Il utilise les notions de Teinte, Saturation et Luminance de la couleur. Il est défini par un espace en forme d'hexacône (voir la figure 3). La teinte T est mesurée par un angle autour de l'axe L. Elle vaut 0 ° pour le bleu, 120 ° pour le rouge, 240 ° pour le vert. La luminance L correspond à l'axe de symétrie de l'hexacône et s'exprime comme la saturation, en pourcentage (de 0 % à 100 %).

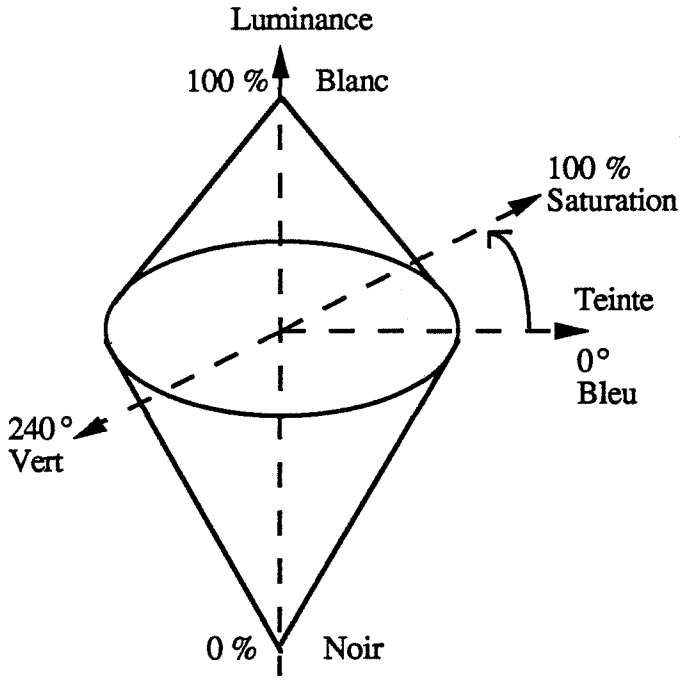


Figure 3 : Le modèle TSL

Les modèles basés sur l'expérimentation intuitive de la perception de la couleur (Teinte Saturation Intensité, ...), donc en particulier TSL, s'avèrent être plus faciles à utiliser. Dans le cas de la production artistique d'images de synthèse, l'utilisateur doit en effet souvent construire intuitivement sa couleur (par l'intermédiaire d'une palette par exemple). Prévoir la décomposition en RVB de la couleur souhaitée apparaît alors comme quelque chose de peu naturel, difficile. Avec le modèle TSL, le processus de création d'une couleur se trouve être simplifié : la teinte, la saturation, la luminance sont des données qui correspondent à nos habitudes culturelles et dont nous savons nous servir .

La visualisation sur un moniteur nécessite cependant de prévoir une conversion du modèle utilisé au modèle RVB. De même que pour le système XYZ, cette transformation s'exprime sous forme de matrice.

2.3.2 Un repère de couleur optimal

La discrétisation de la distribution spectrale, une distribution continue, passe par un échantillonnage de celle-ci. Plus le nombre d'échantillons est important, plus on s'approche du spectre et moins d'erreurs sont commises. Cela nécessitera cependant un accroissement des temps de calcul et une description plus complexe des matériaux et des sources lumineuses, afin de pouvoir tenir compte de toutes ces valeurs.

Hall et Greenberg ([HALL 83]) ont expérimentalement démontré, par comparaison avec un échantillonnage effectué tous les nanomètres, qu'un échantillonnage avec neuf

valeurs fournit, lui, des résultats corrects, avec un temps de calcul supérieur de deux pour cents seulement à celui demandé par l'approche trichromatique.

Réduire le nombre d'échantillons tout en minimisant l'erreur est donc possible, Hall et Greenberg en apportent la preuve. Cependant, aucun critère de sélection des longueurs d'onde n'est proposé dans leur article.

Meyer ([MEYE 88]) propose une méthode de sélection des valeurs de longueurs d'onde basée sur la définition et l'utilisation d'un espace de représentation des couleurs optimal, A C1 C2. Un espace de couleur optimal est un repère dont les axes sont orientés de façon à traverser les régions où les composantes de couleur apparaissent le plus fréquemment. Par l'application d'une quadrature gaussienne, le calcul des trois composantes A C1 C2 est simplifié en un calcul de somme. Meyer propose alors une technique de sélection des longueurs d'onde basée sur ces composantes. L'erreur commise peut être minimisée en répartissant, selon certaines règles, les longueurs d'onde sur chaque composante. Les couleurs obtenues par cette méthode sont plus exactes que celles fournies en utilisant des longueurs d'onde choisies avec l'espace XYZ.

2.3.3 Le rendu des couleurs

La visualisation sur un moniteur vidéo nécessite de calculer la luminance de chaque pixel dans le système trichromatique RVB. La valeur associée à chaque pixel est un spectre qu'il faut donc convertir dans le système RVB (le calcul de ce spectre suppose la connaissance du spectre des différents flux lumineux intervenant). Les composants trichromatiques s'obtiennent par intégration sur toute l'étendue du spectre visible.

$$R = 680 \int_{380}^{780} E(\lambda) r(\lambda) d\lambda$$

$$V = 680 \int_{380}^{780} E(\lambda) v(\lambda) d\lambda$$

$$B = 680 \int_{380}^{780} E(\lambda) b(\lambda) d\lambda$$

où $r(\lambda))$, $v(\lambda))$, $b(\lambda))$ sont les courbes de distribution du système,
$E(\lambda)$ est le spectre énergétique qui détermine le rayonnement.

Il n'est bien sûr pas possible d'effectuer une telle intégration. Deux démarches existent :

- l'analyse spectrale :

On choisit un certain nombre de longueurs d'onde λ_i pour échantillonner le spectre visible (voir le paragraphe 2.3.2). La lumière d'une source lumineuse est décrite par sa distribution spectrale ; la couleur d'une source lumineuse est la combinaison des apparences de couleur des longueurs d'onde sélectionnées. La couleur d'un objet (ou plutôt de sa surface) est la combinaison des longueurs d'onde réfléchies ou réfractées : le spectre de réflectance. Le réalisme d'un modèle basé sur ce principe tient dans la précision de la description des énergies émises et dans la prise en compte exacte des lois de la photométrie et de la colorimétrie. Le calcul du spectre de réflectance est complexe, plus particulièrement pour les matériaux métalliques (voir le paragraphe 3.1.4). Généralement le calcul est effectué pour quelques longueurs d'onde afin de construire une table d'interpolation linéaire.

- l'analyse trichromatique :

Trois longueurs d'onde sont choisies, correspondant aux couleurs RVB. Les couleurs des sources lumineuses, celles des objets et les spectres de réflectance des matériaux sont exprimés à l'aide des trois composantes RVB. Les différents modèles d'éclairement utilisés suite à cette analyse auront donc recours à la couleur des sources lumineuses et des objets (une table bleue éclairée par une lampe rouge). Le problème d'échantillonnage de la distribution spectrale ne se pose pas. Cette analyse, moins coûteuse en temps de calcul et donc plus utilisée, est en fait une approximation de l'analyse spectrale ([BOUA 89]). Elle fournit, a posteriori, des résultats moins bons pour des scènes contenant des matériaux conducteurs (métalliques).

3 Les modèles d'éclairement

Nous tenterons tout au cours de ce paragraphe d'insister sur le coté empirique, ou au contraire physique, des modèles proposés.

La description des modèles d'éclairement utilisera les notations suivantes (voir la figure 4) :

$\vec{N}$ est le vecteur normal en un point de la surface ;
$\vec{V}$ est le vecteur unitaire de visée ;
$\vec{L}$ est le vecteur unitaire de direction de la lumière ;
$\vec{R}$ est le vecteur unitaire réfléchi ;
$\vec{H}$ est le vecteur unitaire bissecteur de $\vec{L}$ et de $\vec{V}$;
$\vec{T}$ est un vecteur unitaire transmis.

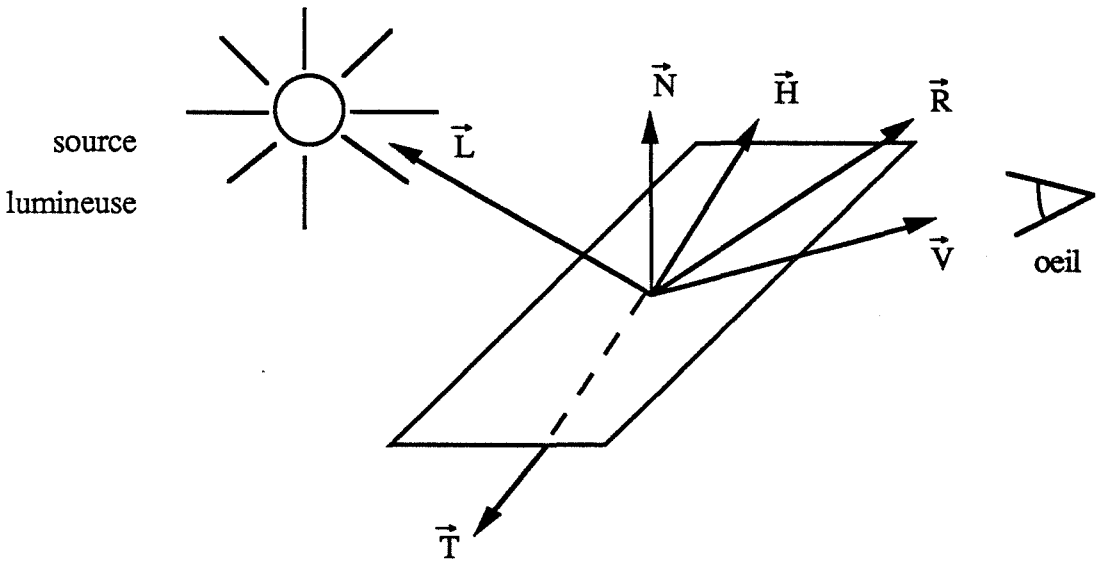


Figure 4 : Notation pour les modèles d'éclairement

3.1 Les modèles d'éclairement locaux

3.1.1 La loi de Lambert

La loi de Lambert spécifie que la luminance perçue par un observateur en un point d'une surface est réfléchie uniformément dans toutes les directions : l'éclairement est dit "diffus". Plus précisement, soit i l'angle du rayon lumineux (ou énergie incidente) avec la normale à la surface ; l'énergie est réfléchie de façon diffuse selon la formule (voir la figure 5) :

$$E_d = K_d * \cos(i) * E_i$$

où

E_i est l'énergie incidente
K_d est un coefficient caractérisant la surface

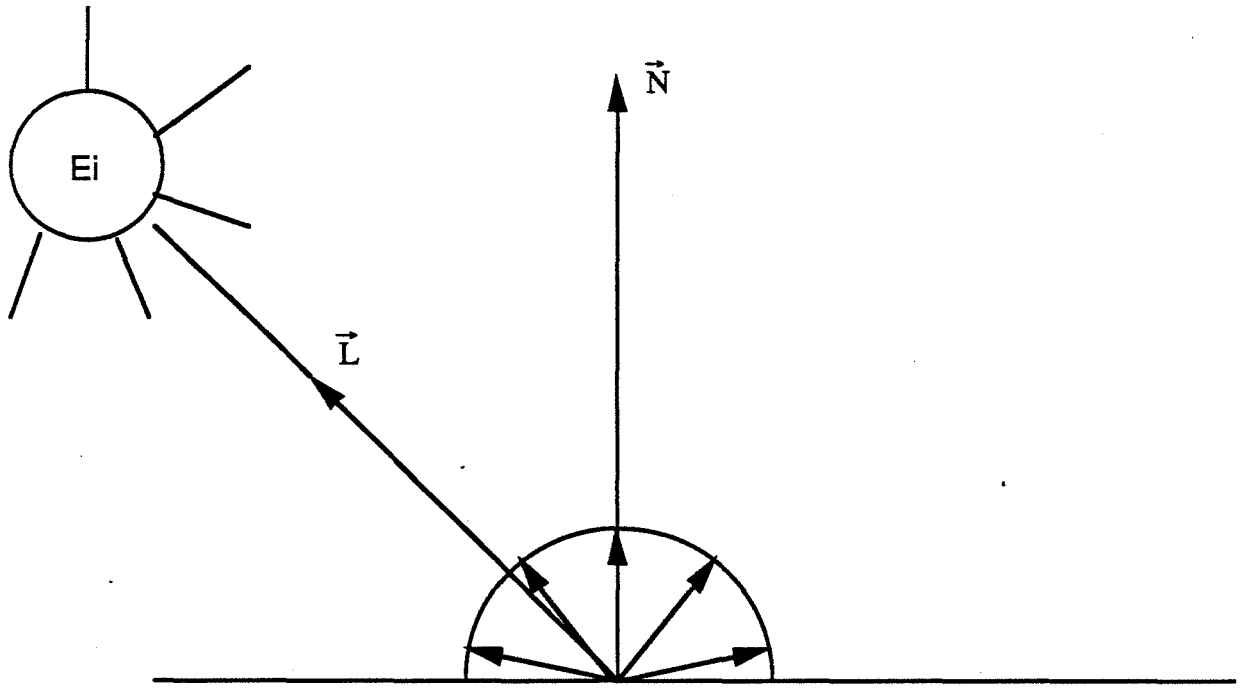


Figure 5 : Réflexion diffuse de Lambert

Pratiquement le calcul du cosinus se fait par un produit scalaire

$$\cos(i) = \frac{\vec{L} \cdot \vec{N}}{\text{norme}(\vec{L}).\text{norme}(\vec{N})}$$

L'éclairement varie d'une valeur maximale, dans le cas où la lampe est située à l'aplomb de la surface (i = 0), à une valeur nulle dans le cas où l'éclairement est rasant, la source étant située dans le plan tangent à la surface (i = 90 degrés). Si le produit scalaire est négatif, la source lumineuse se situe de l'autre coté du plan tangent (i > 90 degrés). Le point est donc considéré comme étant à l'ombre ; il n'est pas éclairé et l'énergie réfléchie est nulle.

A noter que, pour deux faces parallèles éclairées par une source située à l'infini, l'éclairement est le même. Warnock ([WARN 69]), Foley et Van Dam ([FOLE 82]) proposent de tenir compte de la distance d de l'objet à l'œil :

$$E_d = K_d \cos(i)\ E_i\ f(d)$$

$$\text{où } f(d) = \frac{1}{d^2} \text{ ou } \frac{1}{d+k} \text{ avec } k = \text{constante}$$

Basés sur cette loi (le terme de loi me semble plus approprié que modèle), plusieurs modèles ajoutent une composante ambiante, simulant l'illumination globale. Cette composante ambiante est représentée par une constante K_a dépendant de la scène. L'énergie réfléchie par une surface est donc de la forme :

$$E_{ad} = E_a + E_d = K_a + K_d \cos(i) E_i$$

Par rapport à ces modèles, Gouraud ([GOUR 71]) propose une méthode simulant l'aspect de surfaces lisses alors que les surfaces sont modélisées par des facettes planes polygonales. En chaque sommet de chaque facette, un calcul de la courbure (la normale) est effectué. La loi de Lambert est ensuite appliquée en chacun des sommets, pour un calcul de l'intensité. Par un balayage ligne par ligne, une interpolation bilinéaire sur les sommets est effectuée afin de déterminer l'intensité de l'énergie en tout point de la surface. Cette technique très simple permet de simuler des surfaces gauches à partir de polygones. Dans le cas d'une analyse trichromatique, l'appellation "lissage sur les couleurs" est souvent employée pour spécifier, à juste titre d'ailleurs, le lissage de Gouraud.

3.1.2 Le modèle de Phong

Aux modèles basés sur la loi de Lambert, Phong ([PHON 75]) ajoute une autre composante : la composante spéculaire.

La réflexion spéculaire est différente de la réflexion diffuse. Beaucoup de matériaux n'émettent pas effectivement uniformément dans toutes les directions. La réflexion spéculaire dépend aussi de la position de l'œil, ou plus exactement de l'angle s entre la direction de visée et la direction réfléchie (conjuguée par rapport à la normale) du rayon incident (voir la figure 6).

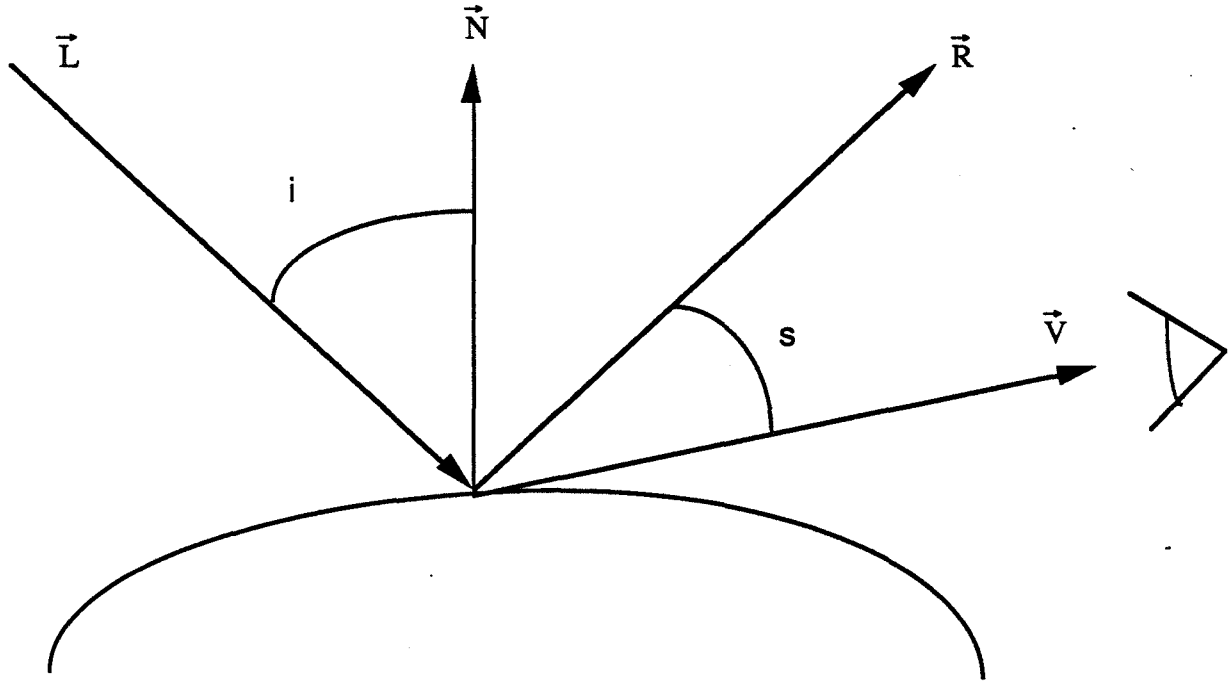


Figure 6 : Réflexion spéculaire

L'amplitude de l'intensité réfléchie est maximale pour un angle nul, l'œil étant situé dans la direction réfléchie (dans le cas d'une surface parfaitement brillante, un reflet est alors perceptible), et décroît au fur et à mesure que l'angle s augmente.

Phong propose la formule suivante :

$$E_S = E_i * W(i) * [\cos(s)^n]$$

où

n est un coefficient de brillance propre à chaque matériau ; il est compris entre 1 et 10

W(i) est un coefficient de réflectance qui dépend de l'angle incident i ; il varie entre les valeurs 10 et 80

Les coefficients W(i) et n expriment les caractéristiques spéculaires de réflexion du matériau ; pour une surface très brillante, leurs valeurs sont élevées.

L'attribution de valeurs à ces coefficients selon le type du matériau est purement expérimentale et ne repose sur aucun modèle physique.

Le modèle de Phong fut cependant le premier à fournir la possibilité de créer des images vraiment réalistes. De par sa relative simplicité, il reste de nos jours encore très employé. La formule générale de l'énergie réfléchie est la suivante :

$$E_r = K_a + [\, K_d \cos(i) + W(i) \cos(s)^n \,] E_i$$

A partir de son modèle, Phong a introduit un lissage différent de celui de Gouraud. Phong interpole les normales en chaque sommet de la surface. Le calcul de l'intensité est ensuite effectuée à l'aide de son modèle. Le cas des surfaces perpendiculaires à la source lumineuse est alors bien traité, ce qui n'était pas possible avec le lissage de Gouraud : l'intensité n'est pas uniforme sur la surface, d'où la notion de reflet. Le lissage de Phong donne des résultats bien meilleurs que celui de Gouraud. Il entraine cependant un accroissement des temps de calculs. Des travaux fournissant des méthodes d'accélération algorithmique ont été proposés ([DUFF 79], [BISH 86], [KUIJ 89]) avec notamment pour objectif de permettre un cablage de ce lissage sur les machines.

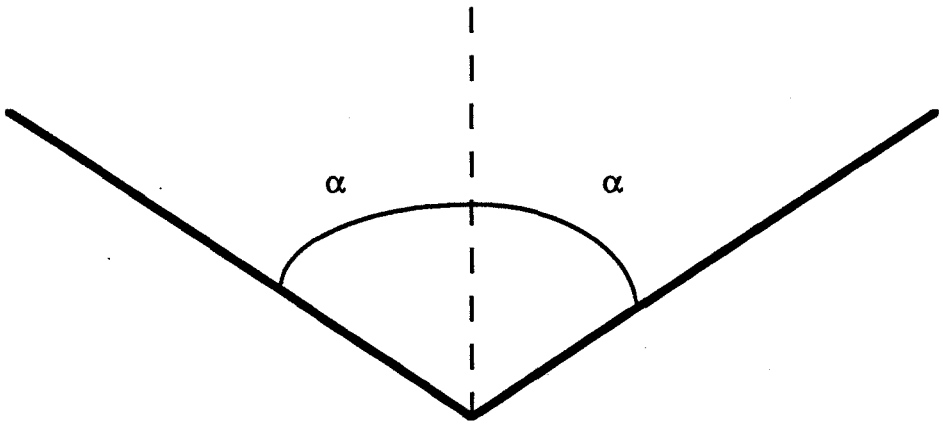
3.1.3 Le modèle de Blinn

Blinn ([BLIN 77]) utilise les résultats de travaux effectués par des physiciens sur la réflectance de la lumière sur des surfaces réelles. En 1967 Torrance et Sparrow ([TORR 67]) ont en effet proposé un modèle théorique décrivant les effets de la réflexion. Toute surface est simulée par un ensemble de facettes microscopiques, parfaitement brillantes. Chaque microfacette est orientée dans une direction aléatoire.

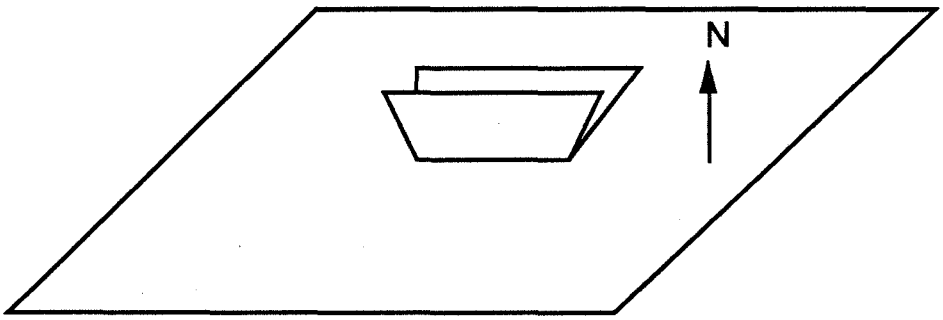
• la composante spéculaire : Torrance et Sparrow introduisent une direction privilégiée pour la réflexion spéculaire, $\vec{H}$, qui est la bissectrice de la direction de visée et de la direction incidente. Seules les microfacettes de normales orientées dans la direction de $\vec{H}$ ($\vec{H}$ constituera donc la normale à ces microfacettes) contribuent à la composante spéculaire. Ces microfacettes sont réparties suivant une loi de probabilité de densité D, qui permet de déterminer la proportion de microfacettes orientées dans une direction donnée. Plusieurs fonctions de distribution de facettes ont été proposées et intégrées au sein de différents modèles (par exemple Blinn et Cook pour ne citer qu'eux).

Torrance et Sparrow émettent un certain nombre d'hypothèses sur les microfacettes :

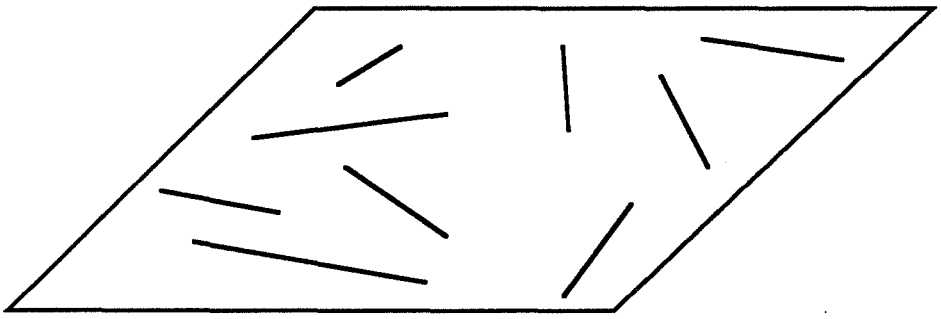
(1) chaque microfacette constitue un coté de la cavité d'une rainure symétrique ;



(2) l'axe longitudinal de la cavité est parallèle au plan de la surface (de normale $\vec{N}$) ;



(3) dans le plan de la surface, les orientations azimutales des axes des cavités sont réparties de façon équiprobable ; le plan contenant le rayon réfléchi et le rayon incident coupe, en général, l'axe de la cavité ;



(4) tous les effets d'ombre et de masquage prennent lieu dans la cavité : les bords supérieurs des rainures sont situés sur un même plan ; aucune microfacette ne dépasse donc les autres ; les deux microfacettes d'une même rainure sont de taille identique ;

(5) à partir de (3) et (4), ils estiment qu'une projection par un plan perpendiculaire à l'axe de la cavité est suffisante pour calculer les effets de masquage et d'ombre ; ce plan contient la normale $\vec{N}$ de la surface et les normales des deux microfacettes.

• la composante diffuse : elle provient des réflexions multiples entre les facettes et de la diffusion interne. Le calcul de la composante diffuse, ainsi définie, semble difficilement réalisable par des méthodes numériques. Ainsi Blinn n'applique ces résultats que pour le calcul de la distribution spéculaire ; il conserve du modèle de Phong les composantes ambiante et diffuse.

La réflexion spéculaire est la combinaison de quatre termes :

$$R_S = \frac{D\ G\ F}{(\vec{N}.\ \vec{V})}$$

• D est la fonction de distribution des facettes : Torrance et Sparrow utilisent, comme fonction de distribution, une simple loi gaussienne

$$D = E^{(-\alpha C_1)^2}$$

où

α est l'angle entre la normale à la surface et $\vec{H}$

C_1 est une caractéristique de la surface

Une surface inclinée par rapport à la direction de l'œil présente plus de microfacettes susceptibles de réfléchir la lumière. L'aire de la surface sera inversement proportionnelle au cosinus de l'angle d'inclinaison, angle entre la normale de la surface et l'œil. Cela explique la division par le produit scalaire, $(\vec{N}.\vec{V})$.

• G est le facteur d'atténuation géométrique : une facette peut se trouver partiellement à l'ombre d'une facette voisine. Une autre peut intercepter la lumière réfléchie par sa voisine. G représente la proportion de lumière qui est véritablement réfléchie, en tenant compte de ces phénomènes ; il varie entre 0 et 1.

Blinn suppose que les rayons lumineux incidents sont parallèles entre eux. Compte tenu des différentes positions de la source lumineuse et de l'œil, trois cas peuvent se présenter (voir la figure 7.1)

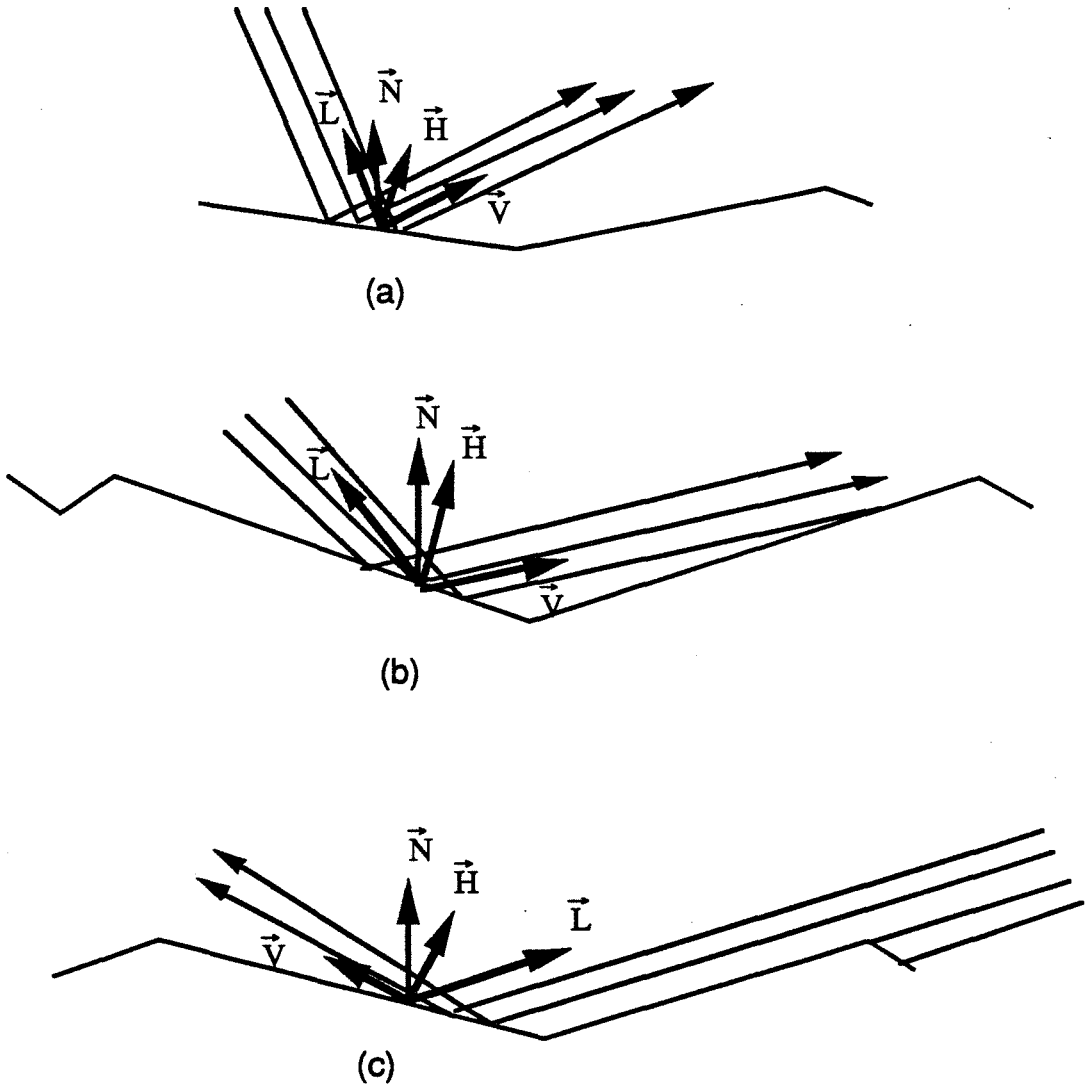


Figure 7.1 : Réflexion sur une surface constituée de microfacettes

cas (a) : La valeur de G est 1 : aucun rayon n'est intercepté, il n'y a pas d'atténuation.

cas (b) : Quelques rayons réfléchis par une facette sont interceptés par une autre. Plus exactement, une proportion de m/l rayons ne parvient pas à l'observateur (voir la figure 7.2). Suivant l'hypothèse (5), nous passons en dimension 2. $\vec{V}$ est projeté dans le plan formé par $\vec{N}$ et $\vec{H}$; $\vec{V}_p$ sera la projection ainsi obtenue (voir la figure 7.3)

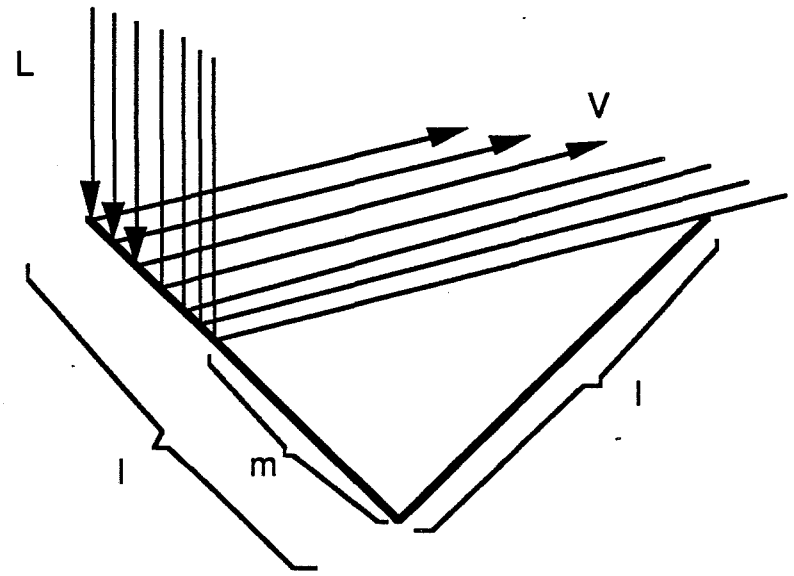


Figure 7.2 : Atténuation géométrique

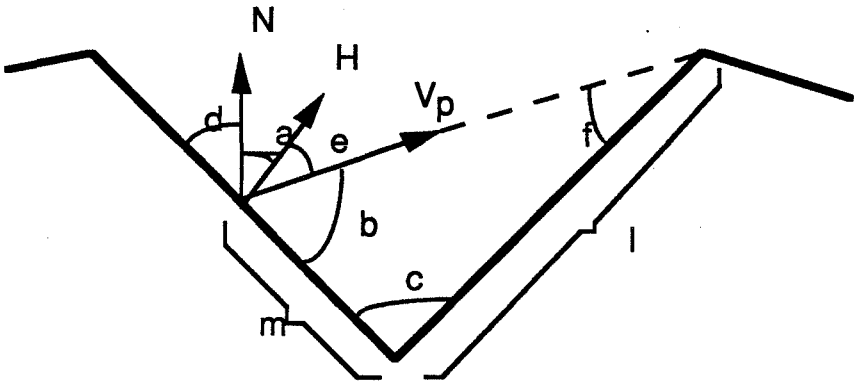


Figure 7.3 : Projection dans le plan

En appliquant la loi des sinus (voir la figure 8)

on obtient $\sin b = \frac{h}{m}$ et $\sin f = \frac{h}{l}$

d'où $\frac{m}{l} = \frac{\sin f}{\sin b}$

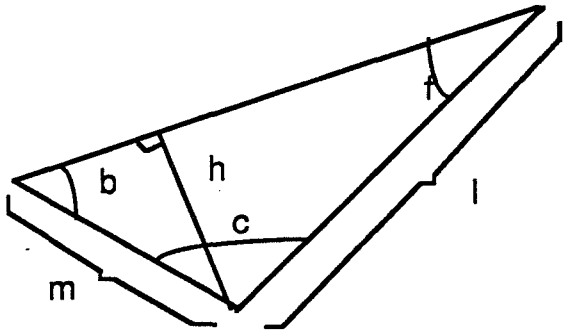


Figure 8

$\vec{H}$ étant la normale à la microfacette, e et b sont complémentaires et ainsi

$$\sin b = \cos e$$
$$\cos b = \sin e$$

La somme des angles d'un triangle vaut π ; on a donc

$$\sin f = \sin (b+ c)$$
$$\sin f = \sin b \cos c + \cos b \sin c \qquad (1)$$

De par la symétrie des rainures (hypothèse (1)) et la complémentarité de a et d, nous pouvons écrire

$$c = 2d$$
$$\cos c = 1 - 2 \sin^2 d = 1 - 2 \cos^2 a$$
$$\sin c = 2 \cos d \sin d = 2 \sin a \cos a$$

En remplaçant dans (1), nous obtenons

$$\sin f = \cos e (1 - 2 \cos^2 a) + 2 \sin e \cos a \sin a$$
$$\sin f = \cos e - 2 \cos a (\cos e \cos a - \sin e \sin a)$$
$$\sin f = \cos e - 2 \cos a \cos (e + a)$$
$$\sin f = (\vec{H}.\vec{V}_p) - 2 (\vec{N}.\vec{H})(\vec{N}.\vec{V}_p)$$

$\vec{V}_p$ étant le projeté de $\vec{V}$ sur le plan formé par $\vec{N}$ et $\vec{H}$,

$$\vec{H}.\vec{V}_p = \vec{H}.\vec{V}$$
$$\vec{N}.\vec{V}_p = \vec{N}.\vec{V}$$

La proportion de rayons non interceptés est donc

$$G_b = 1 - \frac{m}{l} = \frac{2(\vec{N}.\vec{H})(\vec{N}.\vec{V})}{\vec{V}.\vec{H}}$$

cas (c) : Une partie de la lumière incidente est interceptée avant de heurter la surface d'une facette. Seule la surface éclairée de cette facette contribue à réémettre de la lumière. Le calcul de la proportion de surface illuminée est symétrique du cas précédent : il suffit d'intervertir $\vec{L}$ et $\vec{V}$. On obtient donc

$$G_c = 1 - \frac{m}{l} = \frac{2(\vec{N}.\vec{H})(\vec{N}.\vec{L})}{\vec{V}.\vec{H}}$$

Pour une situation donnée, la valeur effective de G sera le minimum de G_a, G_b, G_c, soit

$$G = \min\left(1, \frac{2(\vec{N}.\vec{H})(\vec{N}.\vec{V})}{\vec{V}.\vec{H}}, \frac{2(\vec{N}.\vec{H})(\vec{N}.\vec{L})}{\vec{V}.\vec{H}}\right)$$

• F est le facteur de Fresnel (appelé spectre de réflectance ou spectre de réflexion) : il décrit comment chaque microfacette réfléchit la lumière incidente.

Lorsqu'une onde parvient à la limite entre deux milieux homogènes (d'indice de réfraction n_1 et n_2), elle est séparée en deux ondes : une transmise et une réfléchie. Le calcul du facteur de Fresnel passe donc par le calcul de la réflectance et de la transmittance. Il est basé sur les principes physiques de l'optique. Blinn n'envisage pas de transmission dans son modèle, ainsi intéressons nous uniquement au calcul de la réflectance $\mathfrak{R}$.

Dans le cas d'objets non métalliques, l'énergie d'une onde polarisée réfléchie s'exprime sous la forme (voir [BORN 70]) :

$$\mathfrak{R} = \mathfrak{R}_{//}\cos^2\alpha + \sin^2\alpha\,\mathfrak{R}_{\perp} \qquad (1)$$

où

α est l'angle entre le vecteur électrique E de l'onde polarisée et le plan d'incidence ; il est appelé aussi angle de polarisation.

$\mathfrak{R}_{\perp}$ est la composante de la réflectance perpendiculaire au plan d'incidence

$\mathfrak{R}_{/\!/}$ est la composante de la réflectance parallèle au plan d'incidence

Nous ne considérons que des ondes polarisées. Ainsi nous avons à considérer les valeurs moyennes de $\cos^2 \alpha$ et $\sin^2 \alpha$ pour α entre $+\pi$ et $-\pi$. Ainsi l'équation (1) devient :

$$\mathfrak{R} = \frac{1}{2}[\, \mathfrak{R}_{\perp} + \mathfrak{R}_{/\!/}] \qquad (2)$$

Considérons $R_{\perp}$ et $R_{/\!/}$. Fresnel a introduit un système d'équations appelé "formules de Fresnel" qui définit des coefficients perpendiculaire et parallèle de la réflectance et de la transmittance. Citons le dans le cas de la réflectance :

$$R_{/\!/} = \frac{n_2 \cos \theta_i - n_1 \cos \theta_t}{n_2 \cos \theta_i + n_1 \cos \theta_t} A_{/\!/} \qquad (3\,a)$$

$$R_{\perp} = \frac{n_1 \cos \theta_i - n_2 \cos \theta_t}{n_1 \cos \theta_i + n_1 \cos \theta_t} A_{\perp} \qquad (3\,b)$$

où A est l'amplitude du vecteur électrique de l'onde incidente.

Maxwell a introduit la loi de la réfraction bien connue : (voir la figure 9)

$$\frac{\sin \theta_i}{\sin \theta_t} = \frac{n_2}{n_1} \qquad (4)$$

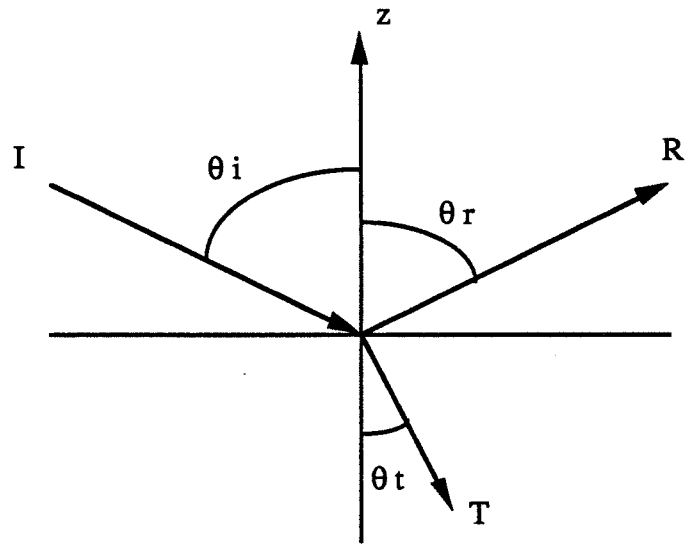


Figure 9

En combinant (4) et (3), nous obtenons :

$$R_{/\!/} = \frac{\tan(\theta_i - \theta_t)}{\tan(\theta_i + \theta_t)} A_{/\!/}$$

$$R_{\perp} = \frac{-\sin(\theta_i - \theta_t)}{-\sin(\theta_i + \theta_t)} A_{\perp}$$

La réflectance est, rappelons le, le rapport de l'énergie de l'onde réfléchie sur celle de l'onde incidente. L'amplitude du champ électrique incident est de la forme :

$$E_i = A \cos (\omega t + \varphi),$$

l'amplitude du champ électrique réfléchi :

$$E_r = R \cos (\omega t + \varphi).$$

En terme d'énergie cela donne :

$$\mathcal{R} = \frac{|R|^2}{|A|^2}$$

d'où

$$\mathfrak{R}_{//} = \frac{|R_{//}|^2}{|A_{//}|^2} = \frac{\tan^2(\phi - \theta)}{\tan^2(\theta + \phi)}$$

$$\mathfrak{R}_{\perp} = \frac{|R_{\perp}|^2}{|A_{\perp}|^2} = \frac{\sin^2(\phi - \theta)}{\sin^2(\theta + \phi)}$$

Blinn rapporte ces travaux et propose comme facteur de Fresnel

$$F(\phi, n) = \frac{1}{2}\left[\frac{\sin^2(\phi - \theta)}{\sin^2(\theta + \phi)} + \frac{\tan^2(\phi - \theta)}{\tan^2(\theta + \phi)}\right]$$

où

ϕ est l'angle d'incidence

n est l'indice de réfraction, qui dépend de la longueur d'onde

$\sin\theta = \frac{\sin\phi}{n}$ (l'indice de l'air vaut 1)

Blinn précise que dans le cas de matériaux métalliques, n prend des valeurs importantes et F se stabilise autour de 1. Dans le cas contraire, n prend des petites valeurs et le calcul de F s'effectue par la formule ci-dessus. Le facteur de Fresnel utilisé est donc bien approprié pour des matériaux non métalliques. Blinn présenta quelques images où l'aspect de rugosité n'atteignait pas un réalisme parfait. Il poursuivit donc son étude sur la visualisation de surfaces rugueuses qui permit ainsi une meilleure simulation de la réalité ([BLIN 78]).

Le modèle de Blinn se distingue de celui de Phong

- par l'apport des facteurs G, F ;

- par la modification de la fonction de distribution des facettes. Blinn estime que, expérimentalement, Phong a proposé comme fonction de distribution un cosinus élevé à une certaine puissance. La fonction de Blinn (celle de Torrance-Sparrow avec simplification) a pour avantage d'offrir de meilleurs résultats avec un calcul plus simple.

Le modèle de Blinn apporte donc encore plus de réalisme. Le temps de calcul, précise-t-il, augmente sensiblement. La formule générale de l'énergie réfléchie est la suivante :

$$E_r = K_a + [K_d \cos(i) + \frac{D\ G\ F}{\vec{N} \cdot \vec{V}}] E_i$$

3.1.4 Le modèle de Cook et Torrance

Les surfaces des objets que l'on rencontre ne sont ni des miroirs parfaits, ni des surfaces diffuses lambertiennes parfaites.

Phong et Blinn ont proposé des modèles dont la réflectance est une combinaison des réflexions spéculaire et diffuse.

Cook et Torrance ([COOK 82]) s'appuient sur les mêmes principes que Blinn mais en permettant de calculer l'éclairement de surfaces rugueuses du type le plus général. Leur modèle de réflexion comporte également trois composantes : ambiante, diffuse et spéculaire.

Considérons une source lumineuse et un observateur. La sensation de luminosité reçue par l'éclairement d'une surface (en d'autres termes, la luminance) varie selon le degré d'inclinaison de la surface par rapport à la direction de l'œil. Physiquement la luminance dépend de l'intensité de l'énergie réfléchie (et non pas de l'énergie incidente, qui elle, est identique) et de la composition spectrale de la lumière réfléchie, toutes deux vues par l'observateur. Le modèle de Cook et Torrance décrit ces deux variations.

L'intensité de la lumière réfléchie dépend de celle de la source, de la surface de la source, de la capacité de réflexion et de la nature de la surface éclairée.

La composition spectrale de la lumière réfléchie est déterminée par la composition spectrale de la lumière émise et par la réflexion de longueurs d'onde sélectionnées par la surface.

Cook et Torrance regroupent le tout en une réflexion appelée réflectance bidirectionnelle que voici :

$$R = s\,R_s + d\,R_d \qquad \text{avec } d + s = 1$$

- R_S ou réflexion spéculaire représente la lumière réfléchie par la surface de l'objet.

Cook utilise la même formule que Blinn, à savoir celle basée sur la théorie des physiciens, mais formulée autrement

$$R_S = \frac{F\ D\ G}{\pi\ (\vec{N}.\vec{L})\ (\vec{N}.\vec{H})}$$

Le facteur géométrique G est exactement le même que celui de Blinn.

Le terme F de Fresnel est légèrement différent dans la mesure où il est valable aussi pour des surfaces métalliques. Physiquement, la formule de réflectance (toujours à partir d'une onde polarisée) fait intervenir un autre coefficient, k, appelé coefficient d'extinction ou indice d'atténuation. L'indice de réfraction n est alors remplacé par n' = n + k. Le coefficient de Fresnel est toujours donné par la relation (2), avec, à partir de (3a) et (3b).

$$R_{/\!/} = \left| \frac{n' \cos \theta i - \cos \theta t}{n' \cos \theta i + \cos \theta t} \right|^2 A_{/\!/}$$

$$R_{\perp} = \left| \frac{\cos \theta i - n' \cos \theta t}{\cos \theta i + n' \cos \theta t} \right|^2 A_{\perp} \qquad (4)$$

Il en découle une formule de Fresnel relativement complexe ([BOUA 89]). Les valeurs de n et k dépendent de la longueur d'onde. Elles ne sont hélas, généralement pas connues. Cependant la réflectance est souvent calculée expérimentalement sous incidence normale. Ainsi, Cook et Torrance proposent :

- si n et k sont connus, on calcule F avec la formule de Fresnel ;

- sinon, si F est connu sous incidence normale,

  - dans le cas d'une surface non métallique, k = 0 et avec l'équation (5) on en déduit n. On peut recalculer ensuite F avec la vraie valeur de l'angle d'incidence ;

  - dans le cas d'un métal, on pose k = 0 et de même, on en déduit n (la valeur expérimentale de la réflectance étant différente pour une surface métallique d'une non métallique, la valeur de n le sera aussi). Le calcul de F est ensuite effectué avec l'équation (5) avec la vraie valeur de l'angle d'incidence.

Toute cette procédure a nécessité la formulation de F (semblable à celle de Blinn) en fonction de n et avec k = 0.

$$F = \frac{1}{2} \frac{(g-c)}{(g+c)^2} \left\{ 1 + \frac{[c(g+c) - 1]^2}{[c(g-c) + 1]^2} \right\} \qquad (5)$$

où $c = \cos(\theta) = \vec{V}.\vec{H}$

$g^2 = n^2 + c^2 - 1$

A incidence normale, nous avons $\theta = 0$ d'où $c=1$, $g=n$. Ainsi

$$F_0 = (\frac{n-1}{n+1})^2$$

et le calcul de n se trouve simplifié, $n = \frac{1 + \sqrt{F_0}}{1 - \sqrt{F_0}}$

Cook et Torrance précisent que les valeurs expérimentales de F sont mesurées sur des surfaces lisses. Il est vrai que le spectre de réflectance des matériaux lisses varie très peu avec l'angle d'indicence ; cette approximation est donc valable surtout pour des objets métallliques.

La fonction de distribution des facettes est la fonction de Beckmann, basée sur la théorie des ondes électromagnétiques des surfaces rugueuses.

$$D = \frac{1}{m^2 \cos^4 \alpha} e^{-[\frac{\tan\alpha}{m}]^2}$$

où α est l'angle formé par la normale à la surface et la direction $\vec{H}$,

m est une caractéristique de la rugosité du matériau : plus le matériau a une surface lisse, plus m est petit (voir la figure 10).

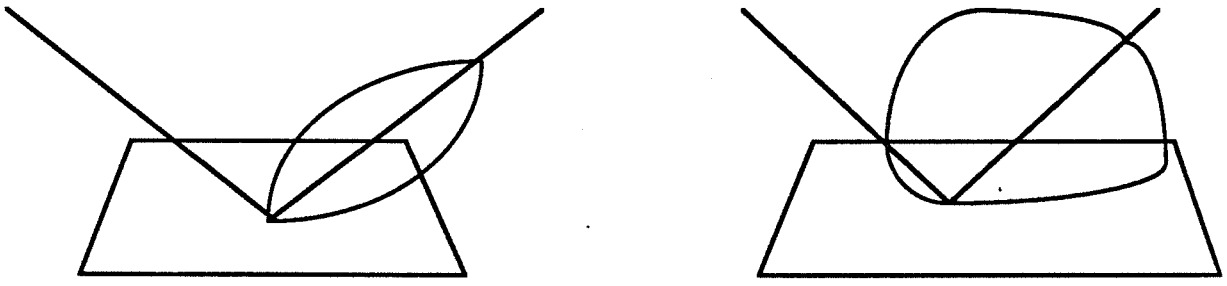


Figure 10 : Fonction de Beckmann pour m = 0.2 et m = 0.6

Ses principaux avantages sont la diversité de matériaux qu'elle peut représenter et la non utilisation d'une constante arbitraire.

Certaines surfaces fortement rugueuses peuvent être caractérisées par plusieurs distributions de pente. Elles sont alors modélisées par plusieurs fonctions de distribution

$$D = \sum_{j=1}^{J} m_j D(m_j) \quad \text{avec} \quad \sum_{j=1}^{J} m_j = 1$$

• R_d ou réflexion diffuse représente la lumière produite par des corps rugueux et mats qui renvoient la lumière uniformément dans toutes les directions. Cela nécessite de tenir compte de toutes les directions où sont renvoyés les rayons lumineux. La contribution finale est alors l'intégrale sur l'angle solide qui contient les rayons réfléchis. Pour simplifier, Cook et Torrance estiment que R_d représente la réflectance bidirectionnelle pour une illumination normale à la surface (pour les métaux, cette approximation est valable). Dans le but de représenter la réflectance des surfaces rugueuses, ils intègrent sur tout l'angle solide, d'où le rapport π :

$$R_d = \frac{F_0}{\pi}$$

Cook et Torrance introduisent également la réflexion diffuse due aux effets d'éclairage indirect, ou illumination globale ambiante. Un coefficient de réflectance ambiant est introduit ; il équivaut à l'intégrale de la réflectance bidirectionnelle.

Ce modèle de réflexion décrit bien le comportement de la lumière réfléchie par toutes sortes de surfaces. Les propriétés électromagnétiques des matériaux sont prises en compte. Le réalisme des images ne peut être que meilleur.

3.2 Les modèles d'éclairement globaux

3.2.1 Le modèle de Whitted

Le modèle de Whitted ([WHIT 80]) sera le premier modèle permettant de calculer l'illumination globale. Il propose (voir la figure 11)

$$E_r = E_a + E_d + K_s S + K_t T$$

où

E_a = énergie ambiante

E_d = énergie diffuse = $K_d \sum_{j=1}^{J} (N . L_j)$

S = énergie spéculaire dans la direction $\vec{R}$

T = énergie transmise dans la direction $\vec{P}$

K_t = coefficient de transmission propre à la surface

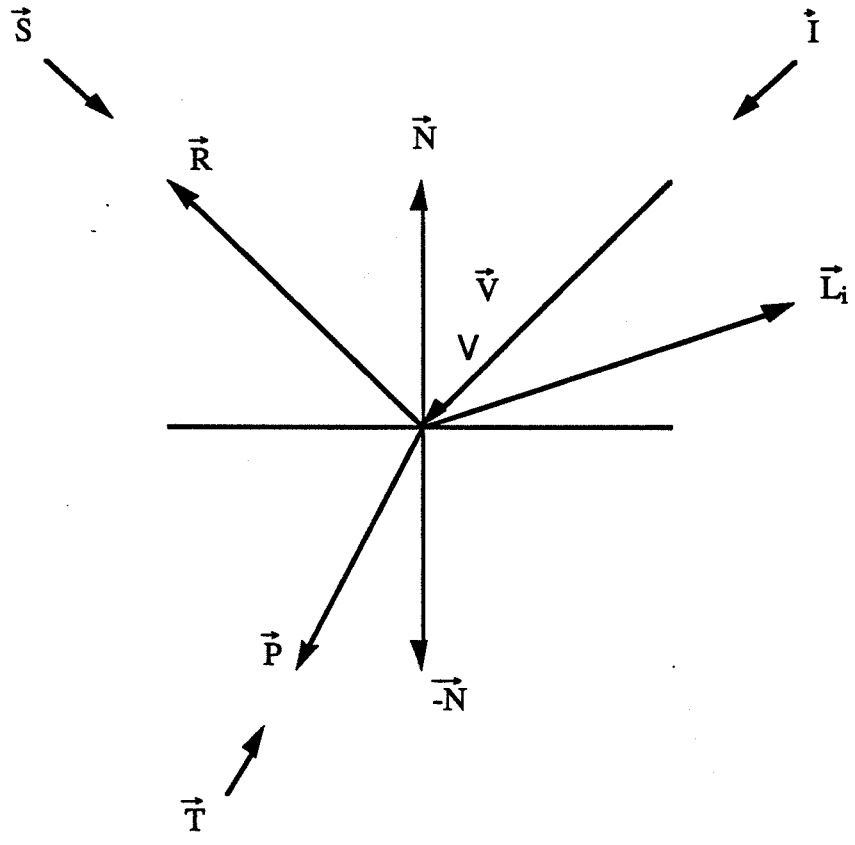


Figure 11 : Modèle de Whitted

Les termes **ambiant** et **diffus** sont repris du modèle de Phong.

La **spécularité** diffère de celle de Phong. Phong tient compte de la spécularité due à la lumière provenant des sources lumineuses. Whitted, dans le but de calculer l'illumination globale, ajoute la spécularité de la lumière due à la réflexion spéculaire des surfaces voisines. Supposons que le rayon provenant de l'œil arrive en une surface 1 (voir la figure 12) ; Whitted cherche à savoir quelle est l'énergie arrivant sur la surface 1 qui provient des surfaces voisines, dans la direction conjuguée R. Considérant maintenant le rayon conjugué, en cas d'intersection avec une surface 2, il faut calculer l'énergie incidente qui part de la surface 2 dans la direction de S_1. Cette dernière dépend également de l'éclairement des autres surfaces voisines. Un troisième rayon est lancé dans la direction S_2. On voit tout de suite l'importante similitude avec l'algorithme du tracé de rayon. Whitted utilisera l'algorithme d'Appel afin d'implémenter son modèle. L'association entre le modèle d'éclairement de Whitted et le tracé de rayon est telle que dans la littérature on emploie souvent l'un à la place de l'autre.

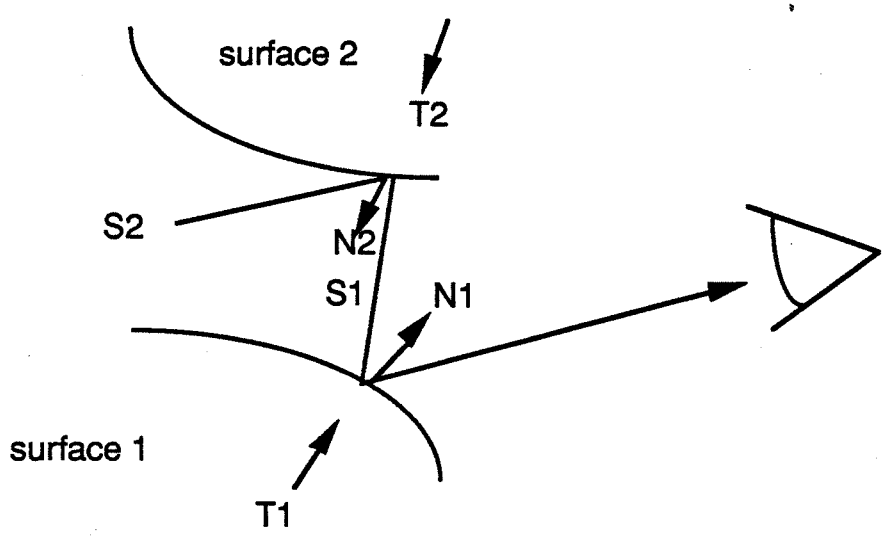


Figure 12 : Suivi inverse du rayon lumineux

Whitted introduit une composante de **transmission** basé sur la loi de Snell. La simulation d'objets transparents est ainsi rendue possible. Toutefois cette transmission se situe le long de la direction T et n'est donc pas réfléchie de façon diffuse à la sortie de l'objet.

Le modèle de Whitted fut le premier, en association avec le tracé de rayon, à proposer des images offrant un réalisme avancé (transparences, ombres portées). Il ignore cependant toute la partie diffusée de la lumière pour ne considérer que la réflexion et la réfraction. Deuxième inconvénient que nous citerons, le coté empirique du modèle qui ne tient pas compte du comportement physique de la réflexion de la lumière comme le font ceux basés sur les travaux de Torrance et Sparrow.

3.2.2 Les modèles de radiosité

A l'opposé du modèle de Whitted, les modèles de radiosité ne considèrent que la partie diffuse de la lumière et non la partie directionnelle. Voyons plus en détail les deux premiers travaux à l'origine de tous les modèles ainsi regroupés dans le terme de radiosité. Nous nous contenterons ensuite d'évoquer leurs successeurs.

3.2.2.1 Le modèle de Goral

Goral ([GORA 84]) permet de mesurer l'intensité diffuse réfléchie à l'intérieur d'un environnement clos. Le principe du modèle est basé sur l'équilibre fondamental de l'énergie : *la totalité de la lumière quittant une surface, ce que nous appelons la luminance, est égale à la somme de la lumière proprement émise par la surface et de la lumière incidente que celle ci réfléchit ou transmet.* Par convention, toutes les surfaces sont considérées comme des surfaces lambertiennes diffuses. Goral propose la relation suivante (voir la figure 13)

$$B_i = E_i + \gamma_i \sum_j B_j F_{ij}$$

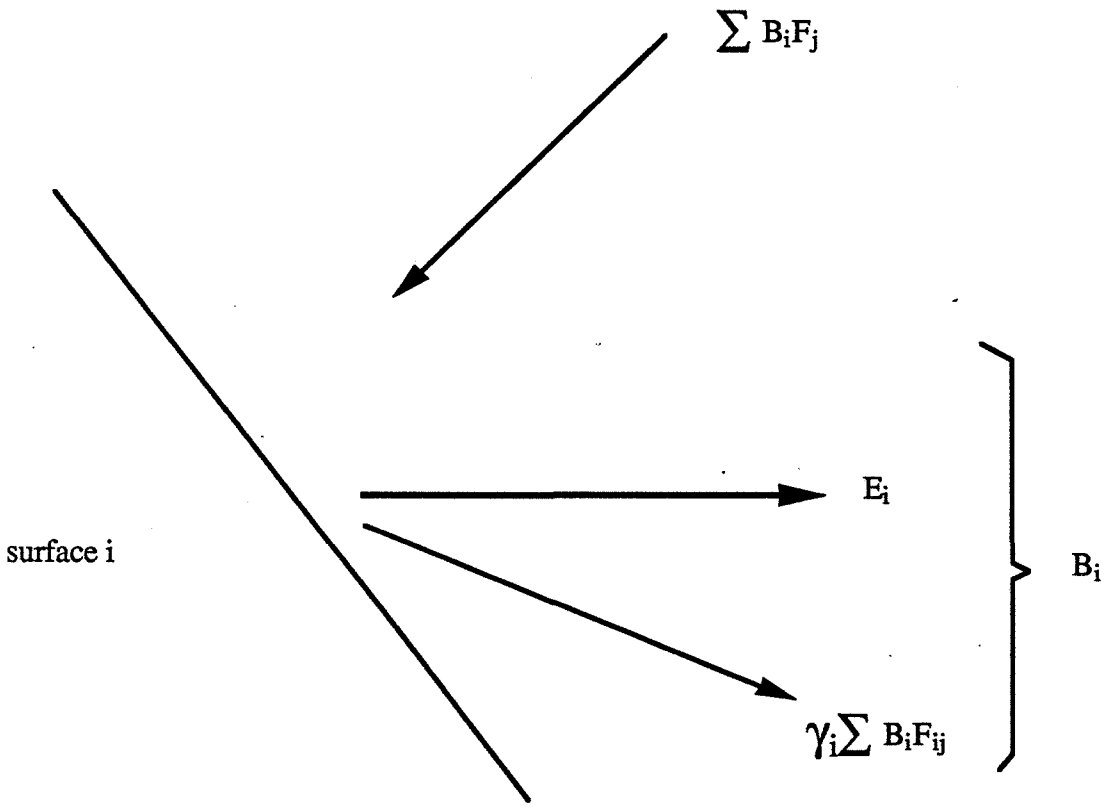


Figure 13 : Radiosité totale

avec les significations suivantes :

$\mathbf{B_i}$ est la quantité d'énergie quittant une surface, soit la somme des énergies émises et réfléchies (unité : énergie/temps/surface).

$\mathbf{E_i}$ est la quantité d'énergie propre émise par la surface i (unité : énergie/temps/surface).

γ_i est la fraction de la lumière incidente arrivant sur la surface i qui est réfléchie de nouveau dans l'environnement (sans unité).

$\mathbf{F_{ij}}$ est la fraction de l'énergie quittant la surface j et qui arrive sur la surface i, appelée plus communément facteur de forme (sans unité).

Si l'environnement est subdivisé en mailles élémentaires (lambertiennes bien sûr) planes, on obtient un ensemble d'équations décrivant l'interaction de l'énergie lumineuse avec l'environnement (voir la figure 14)

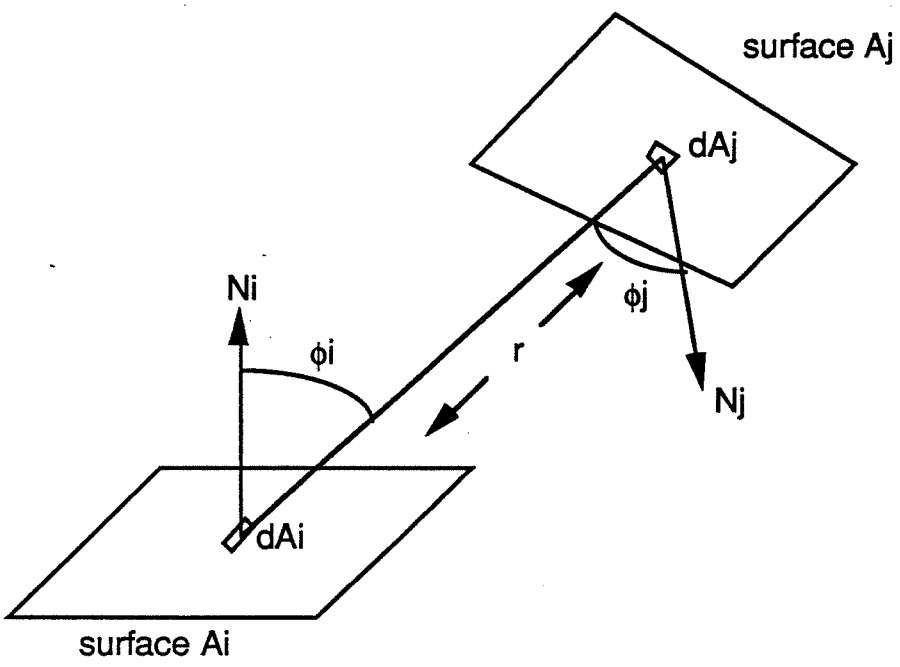
$$\begin{bmatrix} 1-\gamma_1 F_{11} & 1-\gamma_1 F_{12} & -\gamma_1 F_{1n} \\ -\gamma_2 F_{21} & 1-\gamma_2 F_{22} & -\gamma_2 F_{2n} \\ & & \\ -\gamma_n F_{n1} & 1-\gamma_n F_{n2} & 1-\gamma_n F_{nn} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \\ E_n \end{bmatrix}$$

Figure 14 : Système d'équations de radiosité

Chaque objet est caractérisé en outre par sa réflectivité (ou émission dans le cas d'une source lumineuse) dans chaque longueur d'onde du spectre visible. Goral précise bien que

son équation est valable pour des énergies monochromatiques ou possèdant un nombre fini de valeurs de longueurs d'onde. Les termes de réflectivité et d'émissivité sont donc donnés pour une longueur d'onde ou une bande d'onde particulière. La matrice doit être résolue pour chaque bande d'onde de manière à déterminer la luminance complète de chaque maille.

La plus grosse difficulté de calcul concerne les facteurs de forme. Ceux ci s'obtiennent par la formule suivante (voir la figure 15)



$$FA_iA_j = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos \phi_i \cos \phi_j}{\pi r^2} dA_i \, dA_j$$

Figure 15 : géométrie et calcul des facteurs de forme

Goral arrive cependant à simplifier les calculs en se ramenant à une intégrale double sur les contours. Cette méthode, contrairement au modèle de Whitted, propose un calcul de l'illumination globale à partir de la réflexion diffuse des surfaces (Goral emploie le terme de réflexion objet-objet entre surfaces diffuses). Ce calcul tient compte du principe de l'équilibre d'énergie et s'avère plus approprié que celui de Whitted : tout objet réel n'est en général pas un miroir parfait et comporte plus de réflexion diffuse que spéculaire. Quoique très coûteuse en temps de calcul, la méthode de Goral a l'avantage d'être divisée en deux étapes : le calcul des facteurs de forme est effectué en prétraitement indépendamment de la

position de l'œil ; plusieurs vues de la scène peuvent être par la suite calculées en un temps relativement rapide ce qui est appréciable dans une optique d'animation. Le dernier inconvénient concerne les interactions entre les surfaces. Le problème de la visibilité des surfaces l'une par rapport à l'autre n'est pas pris en compte. Le calcul du facteur de forme pour deux surfaces cachées entre elles par une autre surface est effectué. Cependant le cas de deux surfaces dont les normales respectives ont un produit scalaire négatif est quand même traité : leur facteur de forme est nul et l'apport d'éclairement d'une surface à l'autre n'est pas pris en compte.

3.2.2.2 Le modèle de Cohen : l'hémicube

Le dernier défaut de la méthode de Goral a été étudié par Cohen ([COHE 85]) dans le cas d'une scène complexe susceptible de posséder un nombre important de telles surfaces. Par l'intermédiaire d'un hémicube (voir figure 16, tirée de [COHE 85]), le calcul des facteurs de forme se trouve simplifié mais surtout la résolution des faces cachées mutuellement est effectuée. En effet, le facteur de forme d'une surface donnée est approché par la somme des facteurs de forme élémentaires couverts par la projection de cette surface sur l'hémicube. Si deux mailles (ou deux surfaces) se projettent sur un même facteur de forme élémentaire, un tampon en profondeur classique permet de mesurer la profondeur de chaque surface pour le facteur de forme élémentaire considéré, et de résoudre ainsi le problème des faces cachées. L'utilisation d'un hémicube permet de représenter les facteurs de forme élémentaires en pixels carrés. Toute projection de maille se trouve alors simplifiée.

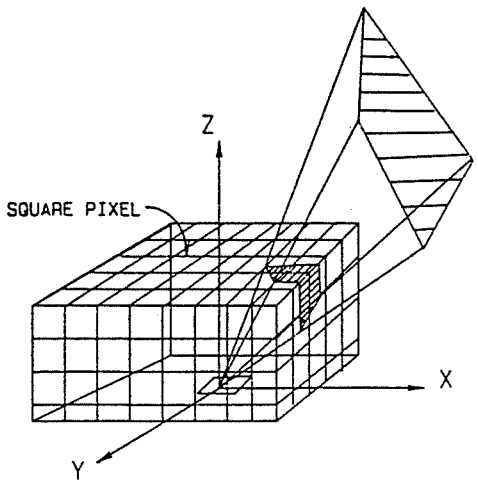


Figure 16 : l'hémicube (d'après [COHE 85])

3.2.2.3 Leurs successeurs

Le modèle de radiosité, nous l'avons déjà précisé, ne considère que la partie diffuse de la lumière. Les effets de spécularité furent donc à l'origine des travaux ultérieurs. Immel ([IMME 86]) propose une méthode de radiosité non totalement diffuse opérant avec cette fois-ci un cube complet (et non un hémicube) orienté toujours dans la même direction. La transparence ainsi que la spécularité diffuse dans toutes les directions sont alors possibles. Des temps de calcul importants sont cependant nécessaires. Shao ([SHAO 88]) propose aussi des possibilités de spécularité à partir d'un hémicube. Des facteurs de forme diffus et spéculaires font l'objet d'un calcul original.

Citons enfin Cohen et al qui ont introduit dans ([COHE 88]) une nouvelle technique de radiosité : la radiosité progressive. Le principe est le suivant : on considère S_1, la surface qui émet le plus de lumière (en terme d'énergie, celle qui émet le plus d'énergie) ; on calcule le facteur de forme de cette surface avec toutes les autres surfaces de la scène ; on détermine la radiosité de chaque surface due à S_1 ; on considère ensuite S_2, la deuxième surface qui émet le plus de lumière ; et on recommence le processus jusqu'à calculer la radiosité complète en chaque surface. Cette technique de calcul progressif de radiosité des surfaces possède deux énormes avantages :

- elle permet de visualiser des images intermédiaires avec des temps de calcul raisonnables (linéaire par rapport au nombre de surfaces ; Cohen et al parlent d'interactivité) ;

- elle nécessite une mémoire de stockage des facteurs de forme très inférieure à la radiosité classique où on calcule les facteurs de forme entre toutes les surfaces.

3.2.3 Les modèles mixtes

L'idée de base de ces modèles est de faire calculer la réflexion diffuse de la lumière par un programme de radiosité et la réflexion spéculaire par un programme de tracé de rayon.

Wallace ([WALL 87]) sera le premier à effectuer un tel mélange. Deux hémicubes orientés comme la face considérée sont utilisés. Le second se trouve être inversé par rapport au premier, afin de tenir compte des transparences. La réflexion diffuse est calculée dans une première étape indépendamment de l'œil par la méthode de l'hémicube. Le calcul de la réflexion spéculaire est ensuite effectué par un tracé de rayon. La réflexion spéculaire est dirigée vers l'œil ; elle est donc à calculer pour chaque positionnement de l'observateur.

Sillion ([SILL 87]) propose aussi un modèle global CIL (Calculs d'Inter_réflexions Lumineuses). Ce système est basé sur l'équation générale de Kajiya ([KAJI 86]), moyennant une hypothèse sur les fonctions de réflectance. Il devient ainsi possible de

modéliser les différents phénomènes de réflexion de la lumière et de visualiser des effets nouveaux en synthèse d'images (la réflexion par un miroir d'un éclairage diffus, par exemple). La complexité générale de la méthode reste cependant inférieure à celle du modèle complet de Immel.

Nous terminerons les modèles globaux en citant le récent modèle global introduit par Bouville et Bouatouch ([BOUV 89]). Ce modèle prend en compte les principes de la photométrie et les indices de réfraction complexes des matériaux conducteurs. Il nécessite alors la connaissance de données physiques telles que le spectre de réflectance, le coefficient d'absorption des surfaces ainsi que le spectre d'émission des sources lumineuses réelles (lampes, soleil, ...). Ce modèle global est donc basé sur les grandeurs photométriques et sur la physique. Il est en fait un système d'équations d'équilibre énergétique similaire au travail de Kajiya, à quelques distinctions près (entre autres, les grandeurs photométriques ne sont pas utilisées par Kajiya contrairement au modèle de Bouville et Bouatouch). Notons que ce modèle est uniquement théorique pour l'instant.

4. Conclusion

La recherche d'un plus grand réalisme en synthèse d'images s'est traduite, au cours de l'évolution des modèles d'éclairement, par deux tendances :

- les modèles utilisent de plus en plus les lois physiques régissant la propogation de la lumière et son interaction avec l'environnement : les derniers modèles proposés prennent en effet en compte des paramètres physiques des surfaces, des aspects photométrique et colorimétrique. On retrouve d'ailleurs souvent ce phénomène en synthèse d'images. Les techniques d'animations font par exemple appel aux lois de la mécanique et de la cinétique. Les derniers modèles de plantes ou d'arbres simulent les lois de la botanique, ...

- un modèle d'éclairement global s'avère indispensable lorsqu'un certain réalisme est exigé. La partie diffuse de la lumière est en effet primordiale, surtout pour une scène comportant plusieurs objets. A ce titre, le modèle de radiosité s'est imposé. L'ajout des effets de spécularité et de transparence a fait l'objet de beaucoup de recherches. L'intégration du modèle de tracé de rayon au sein du modèle de radiosité en est un exemple. Le modèle de réflexion spéculaire du tracé de rayon n'est cependant pas suffisant. Des études sont toujours en cours.

  En restant dans le domaine de la radiosité, citons encore une fois les travaux de Cohen et al ([COHE 88]) sur la radiosité progressive. Cette nouvelle approche (on peut visualiser des images intermédiaires) ainsi que la technique de calcul sont très encourageants pour l'avenir. Qui aurait imaginé en 1984, à la suite des travaux de Goral, qu'on visualiserait interactivement des effets de radiosité ?

Chapitre 2

Les Sources de Lumière en synthèse d'images : notre étude

1. Introduction

Depuis une dizaine d'années, beaucoup de recherches sont menées en synthèse d'images afin de produire des images de plus en plus réalistes. Dans cette quête du réalisme, les ombres portées, les effets de transparence et de réflexion, la génération de textures et l'antialiassage ont été étudiés et des solutions plus ou moins satisfaisantes ont été proposées pour ces divers problèmes ([PERO 88]). Dans ce chapitre, nous allons nous intéresser au problème des sources lumineuses. Résoudre ce problème est un point important pour obtenir des simulations réalistes de scènes dans lesquelles on souhaite représenter des spots de couleur, des lampes de bureau, des phares de voiture, des lampadaires, ...

Notre objectif est de définir des sources lumineuses directionnelles, qui éclairent donc une zone de l'espace définie par le type de source considéré (cône, cylindre ou prisme). De plus, comme dans la réalité, le faisceau lumineux doit être visible (ceci est dû à la présence de grains de poussière dans l'air) tout en étant transparent.

Nos travaux consisteront d'une part à définir un modèle de sources de lumière, et d'autre part à proposer un algorithme de simulation de l'épaisseur de poussière en suspension dans l'air, à l'intérieur de la source lumineuse. Ils seront intégrés au sein de notre système interne de synthèse d'images, ILLUMINES, centré sur un modeleur de type arbre de construction. L'algorithme de visualisation choisi sera le tracé de rayon.

Il faut cependant noter que notre modèle est empirique et que, s'il cherche à simuler la réalité, il ne repose sur aucune modélisation physique de cette réalité.

Dans le paragraphe 2, nous rappellerons quelles sont les principales caractéristiques physiques des sources lumineuses. Nous détaillerons dans le paragraphe 3 les principaux travaux existants pour la réalisation de sources lumineuses en synthèse d'images. Nous les décrirons en les re-situant dans leur contexte (modèle géométrique, modèle d'élimination des parties cachées et modèle d'éclairement utilisés). Nous développerons notre méthode ([FERT 89a]) dans le paragraphe 4 avant de fournir des résultats dans le paragraphe 5 et de conclure dans le paragraphe 6.

2. Les sources lumineuses

La modélisation d'une source lumineuse nécessite essentiellement la définition de trois caractéristiques physiques : la géométrie de la source lumineuse, la distribution de l'intensité lumineuse de cette dernière et la distribution spectrale de l'énergie émise.

2.1 Géométrie de la source

Suivant Greenberg et Verbeck ([GREE 84]), on peut distinguer cinq types de géométrie des sources lumineuses :

- Les sources directionnelles ; ce type de source est le premier à avoir été introduit en synthèse d'images, pour simuler le soleil. La source est définie dans ce cas par une valeur d'intensité et une direction, ce qui permet de spécifier les rayons lumineux parallèles issus du soleil (ou de n'importe quelle source lumineuse située à l'infini) ;

- Les sources ponctuelles ; dans ce cas, la source est spécifiée par un point dans l'espace du monde ; elle est supposée émettre des rayons lumineux dans toutes les directions. Ce type de source correspond, dans la réalité, au cas d'une lampe sphérique nue ;

- Les sources linéaires ; ce type de source, décrit dans [HALL 83] ou dans [NISH 85] définit la source de lumière comme constituée d'un segment de droite ; ces sources sont destinées à modéliser les tubes néon par exemple ;

- Les sources surfaciques ; on trouve dans la littérature deux types de telles sources. Dans [HALL 83], les auteurs mentionnent l'existence de sources définies par une surface de révolution dans leur prototype de test. Dans [BROT 84], la source lumineuse est décrite par un polygone défini dans le repère du monde. Dans ce cas, la source est simulée par un ensemble de points choisis sur la surface, chaque point se comportant comme une source ponctuelle. Ces points peuvent être générés soit par la donnée d'un maillage sur la surface, soit de manière aléatoire. La dernière méthode est préférable, car la première a tendance à causer des effets de zonage sur les parties éclairées par de telles sources surfaciques. Le nombre de points générés doit être proportionnel à l'aire de la source et calculé de telle sorte que l'ajout d'un point ne modifie pas l'éclairement obtenu précédemment ;

- Les sources volumiques ; ces sources, décrites dans [BROT 84] et [NISH 85], sont définies par des polyèdres dont certaines des faces constituent une source lumineuse surfacique ; ces faces sont alors modélisées comme indiqué ci-dessus.

2.2 Distribution de l'intensité lumineuse.

Le modèle le plus classique suppose que la source lumineuse émet uniformément dans toutes les directions. Cependant, ce modèle ne permet pas de simuler la réalité, où les sources peuvent émettre dans une direction privilégiée (à l'aide de réflecteurs) ou dans un secteur de la sphère englobante (à l'aide de diffuseurs).

Pour obtenir plus de réalisme, il faut donc pouvoir décrire les variations directionnelles de l'intensité sur la surface de la source lumineuse. Cette distribution peut être vue comme un champ de vecteurs, l'intensité émise dans une direction étant égale à l'amplitude du vecteur dans cette direction.

Dans l'industrie de l'éclairage, au lieu de définir la distribution d'intensité par un champ de vecteurs, on utilise des diagrammes goniométriques (voir la figure 1).

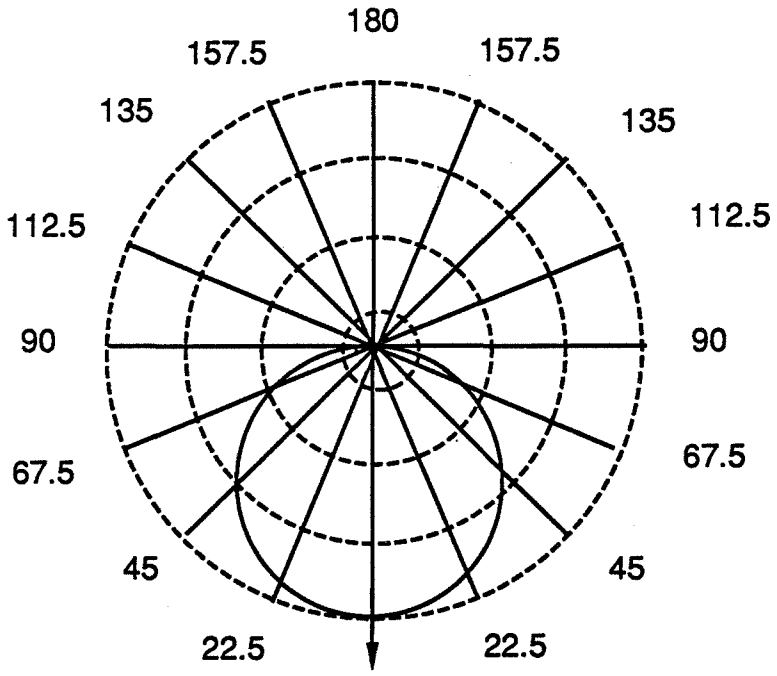


Figure 1 : un exemple de diagramme goniométrique

Ces diagrammes représentent une coupe du champ de vecteurs et définissent ainsi l'intensité en fonction de la direction angulaire. Quand la géométrie de la source et la distribution présentent une symétrie par rapport à un axe, un seul diagramme suffit. Si ce n'est pas le cas (sources linéaires par exemple), on utilise deux courbes, l'une correspondant à l'axe de la source linéaire, l'autre étant perpendiculaire à l'axe de la source ; une interpolation pour les points n'appartenant pas à ces deux plans doit ensuite être effectuée pour déterminer l'intensité en chaque point.

2.3 Distribution spectrale de l'énergie émise

On sait que la lumière est un énergie électromagnétique qui possède une distribution spectrale d'énergie (voir le chapitre 1). La production d'images très réalistes nécessite de fournir la distribution spectrale de la source lumineuse non pas dans le repère RVB, mais à l'aide d'une courbe. La figure 2 présente deux courbes caractéristiques de sources lumineuses particulières (extraites de [HALL 83]).

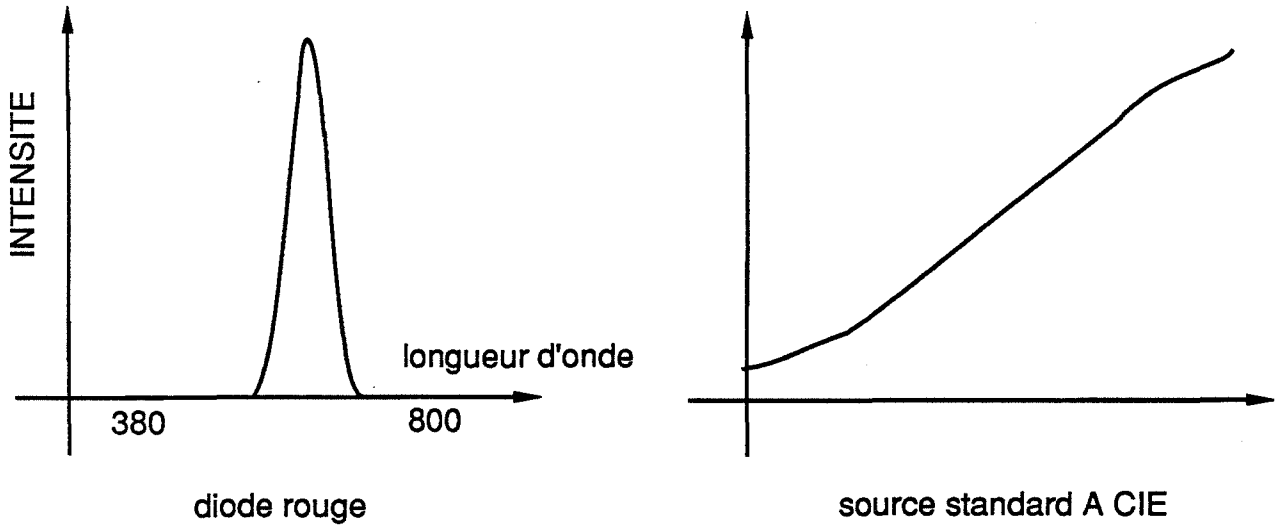


Figure 2 : Distribution spectrale de deux sources lumineuses

3. Présentation des travaux d'autres laboratoires

3.1 Sans visualisation du faisceau lumineux

L'un des premiers qui ait cherché à modéliser des sources lumineuses un peu sophistiquées est Warn ([WARN 83]). Le modèle présenté est intégré dans un système interactif d'aide à la conception d'automobiles, GMR AUTOCOLOR, développé dans les laboratoires de recherche de la General Motors. Ce modèle est essentiellement destiné à simuler des sources lumineuses photographiques, telles qu'elles sont utilisées dans les studios. Le logiciel utilise une représentation polygonale des objets, le modèle d'éclairement de Phong et un algorithme d'élimination des parties cachées du type balayage ligne par ligne. Des sources lumineuses ponctuelles, linéaires, parallèles ou coniques peuvent être définies, dont la direction, la portée et la délimitation sont contrôlées par l'utilisateur de manière interactive. La distribution de l'intensité d'une source lumineuse est également modifiable par l'intermédiaire d'une fonction cosinus élevée à une certaine puissance C. Augmenter cet exposant C revient à concentrer la puissance du faisceau lumineux. Ce procédé empirique permet ainsi de simuler des effets de spot ou, au contraire, de projecteur.

Dans [HALL 83], Hall et Greenberg proposent un système complet, développé à l'Université de Cornell, permettant de créer des images de synthèse réalistes. Ils décrivent un module pour les sources lumineuses, où celles-ci sont définies par les trois caractéristiques exposées au début de ce chapitre. Ainsi, les sources peuvent être ponctuelles, linéaires ou surfaciques de révolution. Par ailleurs, ils visualisent leurs images par l'algorithme du tracé de rayons et ont un environnement de modélisation assez large,

avec possibilité d'utilisation de polygones, d'objets de révolution, de surfaces paramétriques et de quadriques. Enfin, ils ont réalisé un modèle d'éclairement très sophistiqué, basé sur les modèles de Blinn ([BLIN 77]) et de Whitted ([WHIT 80]).

Cook, dans [COOK 84], propose cinq paramètres permettant de définir une source lumineuse sous forme d'un arbre de lumière ; ce sont :

- la position ;

- l'intensité ;

- un paramètre représentant la distance à partir de laquelle l'intensité commence à décroître ;

- un paramètre qui est la distance à partir de laquelle l'intensité est nulle ;

- le type de décroissance (linéaire, exponentielle, ...) de l'intensité entre les deux paramètres précédents.

Dans ([BROT 84]) Brotman et Badler proposent une approche permettant de modéliser les sources lumineuses surfaciques et volumiques distribuées. Le modèle d'éclairement choisi est du type Phong et l'algorithme d'élimination des parties cachées est un tampon en profondeur. Cette approche permet de rendre les effets de pénombre et nécessite alors des temps de calcul ainsi qu'une place mémoire importants.

Tous les modèles ci-dessus ont la particularité commune de ne pas permettre de visualiser le faisceau lumineux lui-même. Le réalisme des images obtenues s'en ressent pour certaines scènes. Les travaux que nous présentons maintenant tentent, eux, d'apporter une solution pour la visualisation des faisceaux lumineux.

3.2 Avec visualisation du faisceau lumineux

Nishita et Nakamae ([NISH 85] et [NISH 87]) sont les premiers à avoir proposé une méthode permettant de visualiser le faisceau lumineux.

Ils peuvent définir des sources délimitées par un cône symétrique ou non par rapport à un axe, et des sources parallèles définies par un prisme. Ils utilisent une modélisation polyédrique des scènes à représenter. La visualisation s'effectue en deux étapes :

- dans une première étape, ils effectuent un pré-calcul (en envoyant des rayons à partir des sources) pour déterminer les volumes d'ombre, de pénombre et d'illumination associés à chaque polyèdre. Pour cela, ils calculent les silhouettes de chaque polyèdre par rapport à plusieurs points de vue

caractérisant la source lumineuse. Ces points sont les extrémités d'un segment si la source est linéaire et les sommets d'un polygone si la source est surfacique. Ils calculent autant de volumes d'ombre qu'il y a de points caractéristiques. L'intersection de tous ces volumes fournit un volume d'ombre global et le plus petit convexe englobant tous ces volumes fournit un volume de pénombre.

• par un balayage ligne par ligne, ils calculent l'illumination des segments en tenant compte

- des différents volumes précalculés auparavant ;

- d'un modèle de distribution de particules, introduit dans [NISH 87], leur permettant de simuler des atmosphères brumeuses ou enfumées à l'intérieur du faisceau lumineux.

Enfin Inakage ([INAK 89]) présente un modèle d'illumination de l'atmosphère. Il faut noter que ce travail a coïncidé avec notre présentation dans ([FERT 89a]).

Sa méthode permet de visualiser les soleils couchants et les effets de brume. Elle est basée sur trois concepts : le cube atmosphérique, l'équation atmosphérique et l'échantillonnage de volume. A partir d'une atmosphère prédéfinie, "le cube atmosphérique", la modélisation de l'interaction de la lumière (absorption, dispersion, ...) à l'intérieur du cube atmosphérique est connue à l'aide de "l'équation atmosphérique", ceci point par point. Or un point de l'espace est occupé soit par un objet, soit par l'atmosphère. Un tracé de rayon permet de résoudre ce problème. Ensuite le calcul de l'éclairement du point d'intersection entre le rayon œil-pixel et la scène dépend de :

- l'éclairement du point considéré par les sources lumineuses à travers l'atmosphère,

- de la réflexion de ce point, en direction de l'œil, toujours à travers l'atmosphère.

Afin d'éviter des calculs d'intégration, un échantillonnage du volume (donc en 3D) atmosphérique original est proposé. L'équation atmosphérique est ensuite appliquée à chaque point de l'échantillonnage.

4. Le modèle proposé

Nos travaux ont été intégrés au sein du système interne de synthèse d'images tridimensionnelles, ILLUMINES ([BEIG 90]), développé par l'équipe Images du département informatique de l'EMSE. Ce système est centré sur un modeleur du type arbre

de construction : CASTOR ([BEIG 86], [BEIG 88]). L'architecture d'ILLUMINES est présentée sur la figure 3.

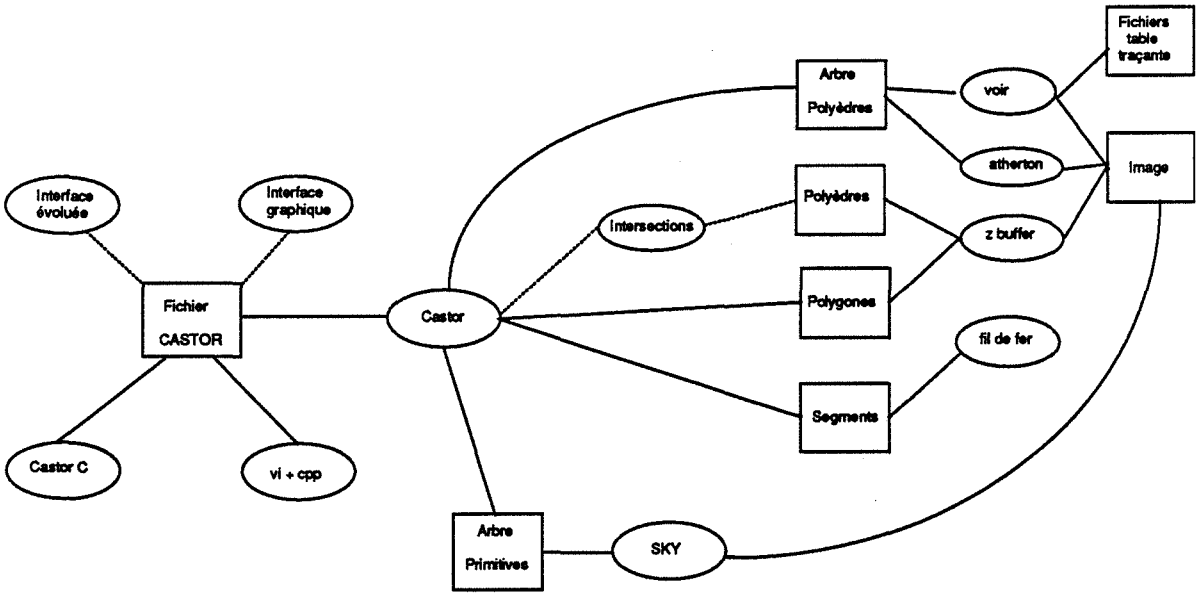


Figure 3 : Schéma récapitulatif d'ILLUMINES

Chacun des cercles sur la figure représente un programme ; chaque carré correspond à un fichier :

castor est l'interprète du langage de modélisation CASTOR ; son entrée est un fichier écrit selon la syntaxe Castor ; il a été écrit par Michel Beigbeder ;

sky est le logiciel de tracé de rayon acceptant en entrée un arbre de primitives ; il a été écrit par Jacqueline Argence ;

voir est l'algorithme d'élimination des parties cachées de Michelucci ; il a été écrit par Dominique Michelucci ;

atherton est l'algorithme d'élimination des parties cachées d'Atherton ; il a été écrit par Dominique Michelucci et Didier Tallot ;

z-buffer envoie les polygones vers la mémoire d'image dans laquelle l'élimination des parties cachées est réalisée grâce à la méthode du z-buffer (microprogrammée sur notre mémoire d'images).

Quand j'ai commencé mes recherches, le laboratoire disposait d'un algorithme de tracé de rayon interne, SKY ([ARGE 88a]). SKY proposait tous les effets de rendu classiques possibles avec la méthode du tracé de rayon (réflexions multiples, transparences, réfraction, ombres portées, textures, ...) permettant ainsi de visualiser des images très réalistes. Seules faisaient défaut les sources directionnelles non ponctuelles, aussi mon travail a consisté à remédier à cette lacune.

Notre objectif de modélisation des sources lumineuses était double : d'une part, introduire des sources de lumière ne rayonnant pas uniformément dans toutes les directions ; d'autre part, nous rapprocher de la réalité en visualisant le faisceau de lumière lui-même et pas seulement son action sur les objets de la scène.

Dans une optique sensiblement identique, Nishita et Nakamae

- limitent la portée du faisceau lumineux par un pré-calcul de volumes d'illumination, de pénombre et d'ombre ;

- introduisent un modèle de distribution de particules à l'intérieur du faisceau lumineux.

Notre méthode

- représente les sources de lumière par des primitives de l'arbre de construction. La connaissance de la portée des sources de lumière, ou plus exactement du volume du faisceau lumineux, élimine tout précalcul.

- utilise un modèle empirique de simulation de poussière. Aucun modèle physique de distribution de particules n'est présenté.

Notre représentation des sources de lumière en primitive "de lumière" de l'arbre de construction possède également d'autres avantages :

- la base de données est uniforme, à savoir exclusivement composée de primitives disponibles du modeleur ;

- nos sources lumineuses sont intégrées de façon naturelle dans la scène et son environnement ; un utilisateur, habitué à la modélisation par arbre de construction, saura les manipuler.

Afin de permettre à l'utilisateur de créer des lampes assez diverses, trois types de sources lumineuses sont proposées :

. les sources coniques pour les lampes de bureau, les lampadaires,... ;

. les sources cylindriques pour les spots de lumières ;

. les sources prismatiques pour l'éclairement provenant d'une fenêtre, d'une porte entrebaillée, ...

Nous allons préciser maintenant les caractéristiques physiques de ces primitives particulières, les procédures de calcul du tracé de rayons permettant de produire les effets souhaités et l'utilisation de ces primitives dans l'arbre de construction.

4.1 Caractéristiques physiques des sources lumineuses

Les sources lumineuses sont susceptibles d'apporter un éclairement de deux façons différentes :

• un objet intercepte le faisceau lumineux et se trouve donc éclairé ; cet éclairement est dit "direct" et la primitive lumineuse est appelée, par distinction avec le second type d'éclairement, "primitive_lumière" ;

• le deuxième éclairement possible est le cas où la source lumineuse est soit devant une primitive, soit seule dans l'espace ; des particules de poussière en suspension dans l'air sont donc éclairées. Ceci nous permet de définir un "éclairement indirect" qui sera calculé en tenant compte de l'épaisseur de poussière traversée (rappelons que Nishita et Nakamae [NISH 87] utilisent la densité de poussière présente dans le faisceau, qu'ils peuvent faire varier, ce qui leur permet de représenter des atmosphères plus ou moins brumeuses ; Inakage calcule, à la suite d'un échantillonnage de volume, l'éclairement de chaque particule échantillonnée par une équation atmosphérique). La primitive de lumière sera, dans ce cas, appelée "primitive_devant". Considérons les trois types de sources lumineuses étudiées :

- dans le cas d'une source conique, l'éclairement produit se calcule à l'aide de deux paramètres : d'une part, la distance entre le sommet du cône et la projection du point d'intersection entre le cône et le rayon œil-pixel sur l'axe du cône (l'éclairement d'un grain de poussière est fonction de sa distance au sommet) ; d'autre part, l'épaisseur de poussière traversée par le rayon, ie la distance entre le point d'entrée du rayon œil-pixel dans le cône et soit le point d'intersection entre le rayon et la primitive, soit le point de sortie du rayon œil-pixel hors du cône (dans ce cas, le cône est dans sa totalité devant une primitive ou seul dans cette zone de l'espace).

- dans le cas d'une source cylindrique, le premier paramètre devient la distance entre la zone émettrice (un disque) de lumière et le point

d'intersection entre le rayon œil-pixel et le cylindre. Le second reste inchangé : l'épaisseur de poussière traversée se calcule de la même façon.

- dans le cas d'une source prismatique, le principe de calcul du premier paramètre est le même ; c'est la distance entre la zone émettrice (un polygone plan) de lumière et le point d'intersection entre le prisme et le rayon œil-pixel. Le second reste également inchangé.

Etudions maintenant pour chacun des types de sources lumineuse créés, comment est calculé l'éclairement direct :

• Les sources coniques sont définies par une primitive du type cône circulaire. L'éclairement produit par ce type de source se calcule aussi à l'aide de deux paramètres : le premier est la distance entre le sommet du cône et la projection du point d'intersection entre le cône et le rayon œil-pixel sur l'axe du cône ; le second est la distance entre le point d'entrée et le point de sortie du rayon œil-pixel dans le cône.

Dans le cas où l'on voit la base du cône, un calcul particulier est nécessaire : l'éclairement doit être concentrique, décroissant du centre du faisceau vers le bord de celui-ci.

Il faut noter, par ailleurs, que l'œil peut être situé à l'intérieur d'un cône de lumière, ce qui n'est pas possible avec une primitive normale. Un traitement particulier s'impose : le rayon œil-pixel traverse une épaisseur de poussière égale à la distance entre l'œil (et non plus la zone émettrice de lumière) et soit le point d'intersection entre la primitive (dans le cas où la primitive de lumière est devant une autre primitive) et le rayon œil-pixel, soit le point de sortie du rayon hors de la primitive de lumière. Les éclairements indirect et direct se trouvent donc être modifiés.

L'objectif de ce calcul d'éclairement direct est d'obtenir un effet correspondant à ce qui se passe dans la réalité, où l'éclairement décroît lorsqu'on s'éloigne de la source de lumière dans la direction d'éclairement, mais aussi quand on s'écarte de l'axe du cône, dans un plan perpendiculaire à celui-ci.

• Les sources cylindriques et prismatiques correspondent à des sources lumineuses directionnelles non ponctuelles. Le rayonnement émis par ces sources est parallèle à l'axe de la primitive lumineuse.

La primitive cylindre est un cylindre à base circulaire. La lumière provient de tout point de la face émettrice (un disque), son intensité est constante sur une face parallèle à la face émettrice et décroît à mesure que l'on s'éloigne de cette dernière. Le calcul d'éclairement direct tient compte, comme pour le cône, de la distance entre le point d'entrée et le point de sortie du rayon dans le cylindre. Le calcul de l'épaisseur traversée est identique à celui effectué pour le cône.

La primitive prisme a pour base un polygone convexe, qui est considéré comme une face émettant la lumière. Le modèle d'intensité est le même que pour le cylindre.

De manière plus précise, le calcul d'éclairement est effectué suivant l'algorithme suivant :

On suppose que les primitives de lumière sont définies dans leur repère local. Prenons l'exemple du cône. Il a pour sommet le point O, pour axe Oz et il est unitaire, c'est à dire que sa hauteur est 1 et que le cercle de base a pour rayon 1. On appellera (voir la figure 8) E le point d'entrée du rayon dans le cône, S le point de sortie du rayon hors du cône, A la projection de E sur l'axe, K (resp. L) le point diamétralement opposé à E (resp. S) dans le plan perpendiculaire à l'axe. On notera E_x, E_y, E_z les coordonnées du point E, d la distance entre E et S, d_1 la distance entre O et A, I l'intensité maximale de la lampe et d_max le maximum des distances entre E et K d'une part, S et L d'autre part.

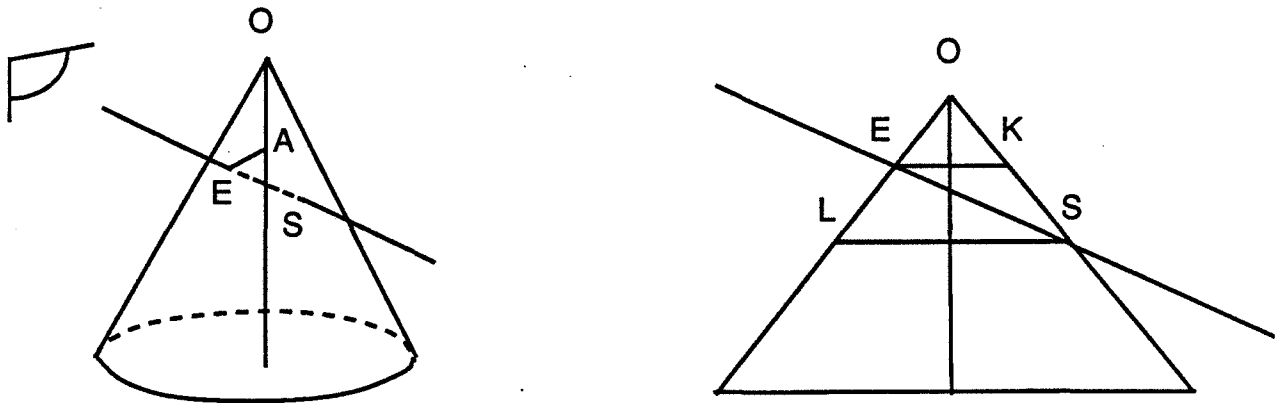


Figure 8

L'algorithme qui suit, détermine l'intensité de l'éclairement pour tout point d'intersection entre le rayon œil-pixel et une primitive de lumière.

Procédure éclairement ();

```
{
    si (Ez= 0 et l'œil est dans le demi-plan z<0) alors
/* on voit la base de la primitive de lumière */
    i := I * (1 - √(Ex²+Ey²)) * (-cos(axe_primitive, axe_visée)) ;
    sinon
    {
        d1:= d(0,A) ;
        i_indirect := 0 ;
        i_direct := 0 ;
        si la primitive est une primitive_devant alors
        {
            si d1 = 0 alors i_indirect := 0 ;
            sinon
            {   si œil à l'intérieur de la primitive alors
                épaisseur_poussière = min(d(œil,point_inter), d(œil,S)) ;
                sinon
                épaisseur_poussière := min(d(E,point_inter), d(E,S)) ;
                i_indirect := I * (1-d1) * épaisseur_poussière ;
            }
        } ;
        si la primitive est une primitive_lumière alors
        {
            si d1 = 0 alors i_direct := I ;
            sinon
            {   si œil à l'intérieur de la primitive alors
                    épaisseur_poussière = d(œil,S) ;
                sinon
                épaisseur_poussière := d(E,S) ;
                d_max := max(d(E,K), d(L,S)) ;
                i_direct := I * (1-d1) * épaisseur_poussière / d_max ;
            }
        } ;
        i := i_direct + i_indirect ;
    } ;
}
```

Les sources lumineuses étant des primitives de l'arbre de construction, on peut les soumettre à des transformations affines (rotation, translation, affinité, ...), ce qui augmente la diversité des lampes possibles. Elles possèdent également une couleur, qui est spécifiée dans le modèle RVB. Pour effectuer les calculs d'éclairement présentés dans l'algorithme ci-dessus, nous passons dans le modèle TSL. Nous ne considérons des sources lumineuses que leur luminance, que nous assimilerons à l'intensité de la lampe. A l'aide de l'algorithme décrit ci-dessus, nous calculons la luminance en chaque point d'intersection. En revenant au modèle RVB, nous avons ainsi l'apport de couleur de chaque source lumineuse. Le modèle d'éclairement utilisé dans le logiciel de tracé de rayon du laboratoire est du type Whitted. Il offre donc en plus de la réflexion diffuse, une réflexion spéculaire pour les sources ponctuelles. Dans une première étape, nous avons apporté la contribution des sources directionnelles de lumière uniquement au niveau de la composante diffuse. Nous envisageons ultérieurement de tenir compte de la spécularité provenant de l'éclairement direct de nos types de source :

- pour la source conique, ponctuelle, la connaissance de la réflexion spéculaire permet d'envisager une telle tentative ;

- pour les sources surfaciques cylindriques et prismatiques, il en est autrement. Nous n'avons pas relevé dans la littérature, de méthodes tenant compte de la spécularité de sources surfaciques, ceci pour n'importe quel algorithme d'élimination des parties cachées. Nous connaissons les travaux de Brotmann ([BROT 84]) sur la détermination des ombres provoquées par de telles sources. Toute source surfacique est considérée comme un ensemble de sources ponctuelles. Un échantillonnage est effectué. Ce procédé peut être utilisé pour notre problème. Mais quelle direction prendra le rayon réfléchi contribuant à la composante spéculaire ? Sera-t-elle la même pour tous les points échantillonnés ? Le réalisme et la complexité trancheront.

4.2 La procédure d'intersection

Dans un premier temps, les primitives de lumière ne peuvent être filles que d'un noeud union. Nous n'autorisons pas d'intersection ou de différence entre une primitive de lumière et une primitive autre que lumière (conceptuellement, à quoi est-ce que cela correspondrait ?), mais également entre deux primitives de lumière (nous préciserons, en conclusion de ce chapitre, les éventuelles extensions à ce sujet).

Grâce à un marqueur qui permet de la distinguer d'une primitive ordinaire, la primitive de lumière possède les propriétés suivantes :

- elle est transparente. Tout objet situé derrière la primitive de lumière sera donc vu ; cependant, l'existence des particules de poussière en suspension dans l'air se traduit par un apport de couleur, correspondant à celle de la

primitive de lumière (c'est l'éclairement indirect étudié dans le paragraphe 3.2) ;

- comme toute primitive, une primitive de lumière située derrière un autre objet (non transparent) ne sera pas vue ;

- si deux primitives de lumière se coupent, l'éclairement de la zone commune est calculé en ajoutant l'éclairement relatif à chacune des primitives.

Pour mener à bien les calculs d'éclairement définis dans le paragraphe précédent, nous devons, pour chaque rayon, connaître les points suivants :

- le point d'entrée du rayon dans la primitive de lumière, point_1 ;

- le point de sortie du rayon hors de la primitive de lumière, point_2 ;

- le point d'entrée du rayon dans le premier objet vu, point_3.

Si point_1 < point_3 < point_2, il y a intersection entre la primitive de lumière et l'objet.

Si point_1 < point_2 < point_3, on visualise l'objet non éclairé, mais légèrement illuminé par le faisceau qui est situé entre l'œil et lui.

Si point_3 < point_1 < point_2, l'objet est visualisé normalement et la primitive de lumière est cachée par l'objet.

4.3 Utilisation de l'arbre de construction

L'utilisation d'un arbre de construction pour modéliser une scène visualisée par tracé de rayons est un procédé classique. Nous allons maintenant présenter les modifications imposées par l'adjonction de primitives de lumière au modeleur.

Nous avons été amenés à gérer trois listes d'enregistrements :

- La première liste est celle des primitives de lumière éclairant le point d'intersection considéré. Les primitives sont rangées dans la liste par distance croissante entre l'œil et le point d'entrée du rayon dans le cône. Chaque enregistrement de cette liste contient :

  • les caractéristiques de la primitive :

. son origine dans le cas du cône, ou le point de la base émettrice dans le cas du cylindre et du prisme ;

. sa couleur ;

• le point d'intersection du rayon avec la primitive ;

• la distance entre l'œil et le point de sortie du rayon hors de la primitive ;

• la primitive suivante dans la liste.

- La seconde liste est constituée des primitives lumineuses situées devant le point d'intersection considéré (elles sont appelés primitives_devant). Chaque enregistrement possède les mêmes champs que ceux de la liste précédente ; les primitives sont rangées dans la liste comme pour la première liste.

- La troisième liste contient les primitives lumineuses, nommées "primitive_dedans", qui sont coupées par le rayon passant par le point d'intersection considéré. Cette liste est rendue nécessaire par la présence des opérateurs booléens intersection et différence dans l'arbre de construction : ces deux opérateurs peuvent rendre des parties de l'espace éclairées par une primitive de lumière, alors que ce n'était pas le cas avant de résoudre l'opération booléenne.

Enfin, du fait de la propagation de la lumière, un objet situé derrière un autre par rapport à une primitive de lumière ne pourra être éclairé par cette dernière.

Ainsi, il ne faut pas :

- que la partie de l'espace située derrière le premier objet rencontré soit éclairé ;

- que les objets situés derrière le premier objet par rapport à la source de lumière soient éclairés par ladite source de lumière.

Pour cela, il faut lancer un rayon secondaire

- de point_3, intersection entre le rayon primaire et l'objet, vers le point émetteur de lumière (l'origine du cône, par exemple), dans le cas d'une primitive de lumière ;

- de point_1, intersection du rayon primaire avec le cône, vers le point émetteur de lumière, dans le cas d'une primitive_devant.

S'il y a intersection entre le rayon secondaire et un objet autre qu'un cône de lumière, on en déduit que l'objet considéré n'est pas éclairé par le cône.

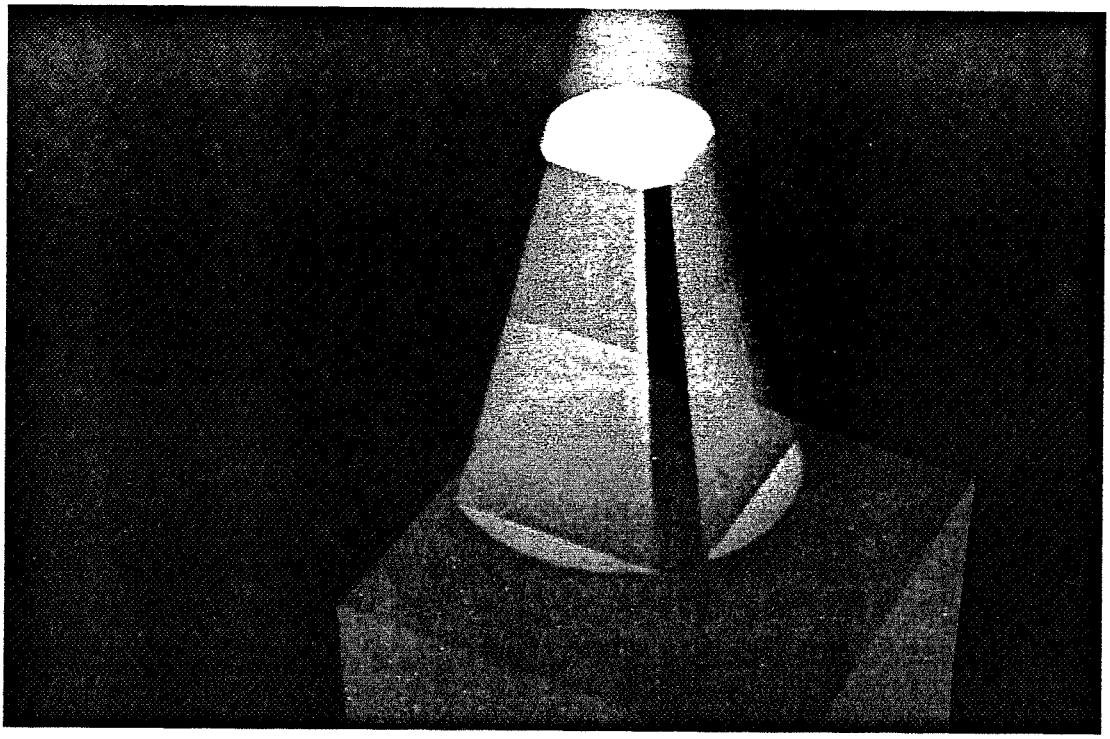


Image 1 : Cette image s'efforce de mettre en évidence la simulation de la propagation de la lumière et de poussière. Un faisceau de couleur blanche éclaire (très fortement d'ailleurs, l'ajout de couleur blanche atteint alors vite le maximum 255 255 255, d'où la couleur blanche) un cube rouge. Ce dernier coupe en partie seulement le faisceau lumineux, ainsi :

- le cube inférieur vert est éclairé uniquement par la partie du cône de lumière qui n'a pas été interceptée par le cube rouge ;

- les particules de poussière illuminées en suspension dans l'air, situées devant le cube rouge et devant le cube vert, apportent également un surplus de couleur, plus faible cependant que celui provoqué par l'éclairement direct.

4.4 La résolution des problèmes d'aliassage

Une image de synthèse peut souffrir de défauts d'aliassage. Ces phénomènes proviennent d'un échantillonnage insuffisant lors du processus de visualisation. Les méthodes usuelles d'antialiassage du tracé de rayon consistent, le plus souvent, à sur-échantillonner globalement l'image (par exemple, chaque pixel est divisé en quatre sous-pixels). Les pixels aliassés ne constituent qu'une faible partie de l'image, aussi un sur-

échantillonnage local des pixels présentant un problème, est-il plus intéressant, notamment sur le plan des temps de calculs. On peut citer les travaux de Whitted ([WHIT 80]) qui mesure les gradients d'intensité sur quatre pixels voisins, ou ceux de Roth ([ROTH 82]) qui compare les objets vus en deux pixels adjacents.

La reconnaissance et le nombre des pixels aliassés diffèrent selon les méthodes. Dans [ARGE 88b]), Argence a proposé une méthode originale de résolution du phénomène de l'aliassage dans le cadre d'une modélisation par arbre de construction. Cette solution consiste en un sur-échantillonnage local et adaptatif des pixels présentant un problème. Cette méthode nécessite plusieurs informations en chaque pixel :

- le numéro de la primitive interceptée ;

- le numéro de la surface C1 interceptée (à chaque surface C1 de chaque primitive est associée un numéro) ;

- la liste des pointeurs sur les objets interceptés

  dans le cas d'un objet *simple* (un noeud dont les ancêtres sont uniquement des unions), un pointeur sur le premier noeud suffit ;

  sinon toutes les primitives considérées sont à garder dans la liste ;

- un drapeau qui indique si la primitive est éclairée par une source lumineuse ou pas ;

- la liste des pointeurs sur les objets *simples* faisant de l'ombre pour chaque source lumineuse.

L'antialiassage, récursif, est effectué de la façon suivante. Chaque pixel est comparé avec ses huit voisins. Un défaut d'aliassage peut apparaître sauf si les neufs pixels vérifient les conditions suivantes :

- ils ont le même numéro de primitive ;

- ils ont le même numéro de surface ;

- ils sont éclairés par les mêmes sources lumineuses ;

- il n'y a ni réflexion, ni réfraction ;

- aucun petit objet n'est testé.

Si une de ces conditions n'est pas remplie, le pixel est divisé en neuf sous-pixels. Pour chacun des pixels sur-échantillonnés, une sous-scène pour les rayons primaires est construite, une autre pour les rayons vers les sources lumineuses, et une autre pour les rayons de réflexion, de réfraction, ... La couleur du pixel considéré est alors la moyenne des valeurs des neuf sous-pixels.

Les possibilités de défauts d'aliassage dues aux sources ponctuelles sont donc prises en compte par la méthode d'Argence. L'ajout de nos sources lumineuses

- a nécessité de notre part une prise en compte des structures mises en place et une nouvelle gestion. Argence utilise une liste des objets interceptés ainsi qu'un drapeau indiquant si une primitive est éclairée. Nous incluons, pour notre part, la primitive de lumière dans la liste des objets interceptés (les primitives de lumières sont des primitives à part entière de l'arbre de construction). Par comparaison avec les autres objets, nous pouvons conclure si la primitive de lumière est susceptible de créer de l'aliassage ou pas ;

- a entrainé deux cas particuliers supplémentaires dus à la visualisation du faisceau lumineux :

  - un pixel de la couleur du fond de l'image est voisin d'un pixel éclairé par une source lumineuse ;

  - un pixel éclairé par une (ou plusieurs) source lumineuse (mais sans visualisation d'un autre objet) est voisin d'un pixel contenant la couleur d'un objet vu à cet endroit ; objet éclairé également par la source lumineuse.

Ce dernier cas pose problème dans la mesure où, pour les deux pixels, la primitive de lumière figure en tête de liste des objets interceptés. Il est alors nécessaire de parcourir cette liste afin de distinguer ce cas particulier.

5. Résultats

Un modèle de source de lumière a été présenté pour visualiser le faisceau lumineux de la source lumineuse. Les résultats obtenus par cette méthode sont d'un réalisme satisfaisant comme le montrent les images qui suivent. Le but principal que nous nous étions fixé a été atteint : générer des images qui provoquent des sensations visuelles identiques à celles provoquées par une scène réelle.

Notre méthode s'intègre de façon naturelle à l'algorithme de tracé de rayon dans le cadre d'une modélisation par arbre de construction : les sources de lumière sont des primitives de l'arbre de construction presque identiques aux autres. Dans le cas où elles

interceptent le rayon œil-pixel, aucune intersection n'est signalée. Un drapeau ainsi que toutes les caractéristiques de la primitive lumineuse permettent de connaître l'existence de la primitive de lumière en ce pixel et à ce niveau dans l'arbre de construction.

La représentation originale des primitives de lumière en primitives de l'arbre de construction résoud le problème de la connaissance de la portée des sources lumineuses. A l'encontre de la méthode de Nishita et Nakamae, aucun pré-calcul n'est effectué.

Le modèle proposé afin de simuler l'existence de particules de poussière en suspension dans l'air à l'intérieur de la primitive de lumière est empirique. Cet algorithme a été conçu pour donner l'impression réelle de poussière mais il ne repose, répètons le, sur aucune modélisation physique de la réalité.

Notre méthode permet de réaliser des sources coniques et parallèles avec effet de faisceau lumineux. De par la simplicité de leur définition, l'utilisateur peut créer aisément des sources de couleurs complexes, de formes diverses et d'intensité variable (la portée de la lampe lui est demandée). Les images qui suivent (visualisées sur un LEXIDATA) en fournissent des exemples.

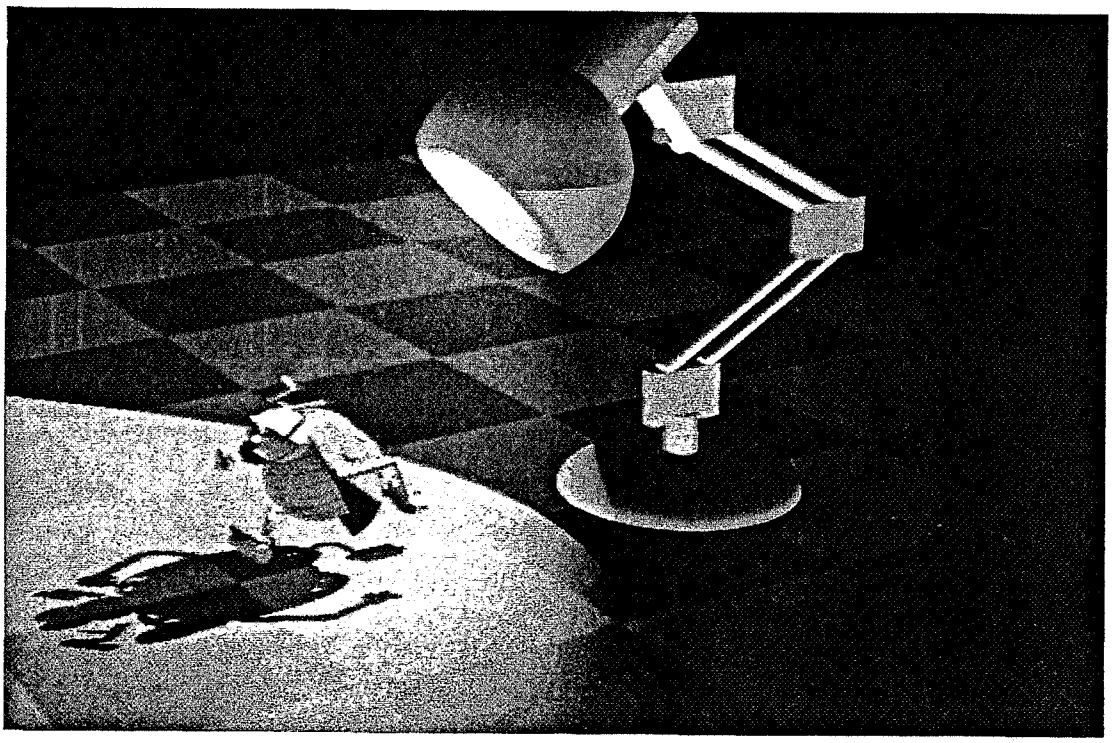


Image 2 : Cette image représente une scène comportant une lampe de bureau obtenue à partir d'une source conique. Le calcul des ombres portées est effectué classiquement par l'algorithme du tracé de rayon. On peut constater la non-uniformité de l'effet de transparence du faisceau, surtout sur le bord de celui-ci. La coccinelle et la lampe ont été conçues par Fabienne Rouault.

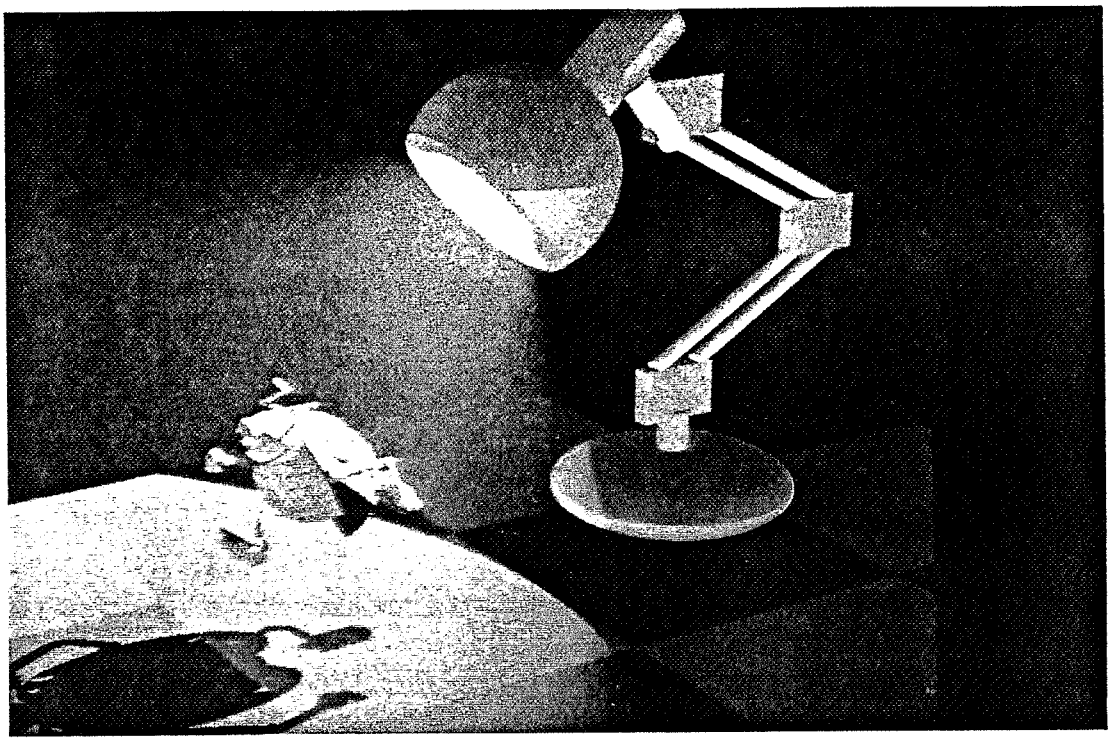


Image 3 : Image identique à la précédente avec un damier plus petit. On distingue mieux alors la variation de l'illumination de l'espace due au faisceau lumineux.

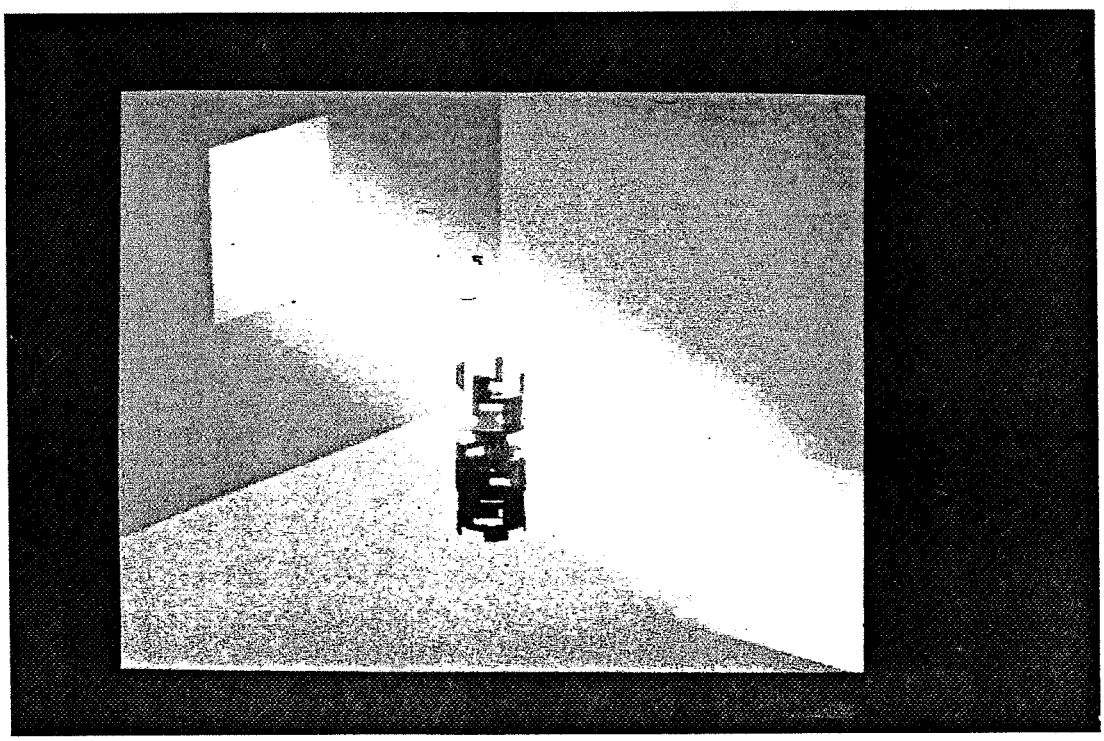


Image 4 : Une source prismatique simule la lumière pénétrant par une fenêtre dans la pièce. Un cylindre creux situé au milieu de la pièce est éclairé et

permet de constater l'intégration de notre modèle au sein d'un arbre de construction, comprenant les opérations booléennes différence et intersection. L'ombre du cylindre se reflète sur le sol. Le cylindre creux a été conçu par Gabriel Hanotaux.

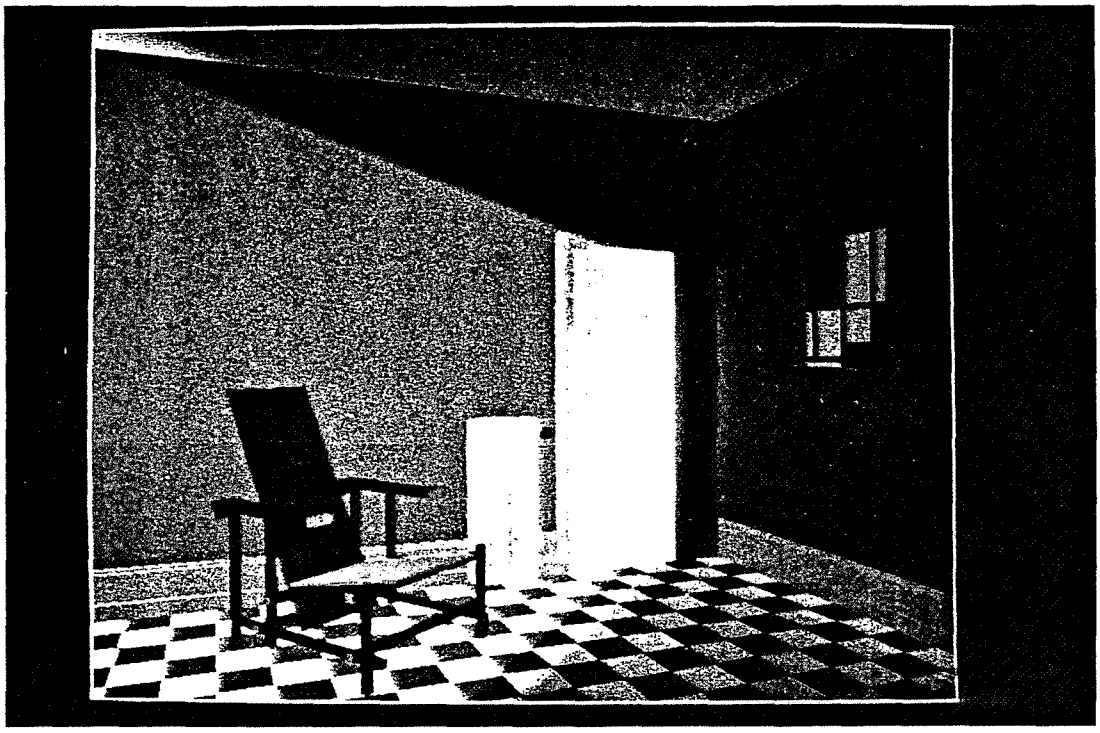


Image 5 : Conception Philippe Jaillon. Une autre source prismatique est utilisée pour simuler un entrebaillement de porte. Cette image fait apparaître une lacune de notre méthode. Aucune ombre, proche des pieds de la chaise, n'est visualisée. Deux raisons expliquent cette absence (voir la figure 9) :

- Nos sources surfaciques sont directionnelles. Elles sont considérées comme constituées par un ensemble de sources ponctuelles émettant uniquement dans la direction parallèle à l'axe de la primitive de lumière.

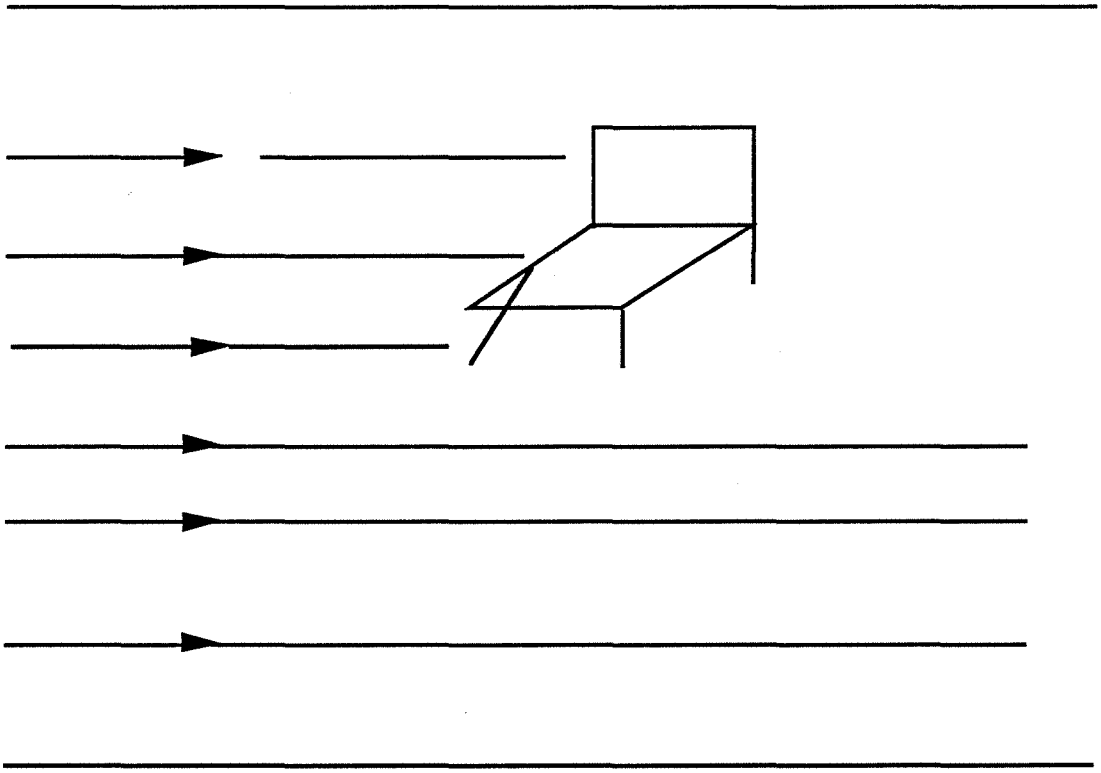


Figure 9 : Sources directionnelles parallèles au sol

Elles sont très représentatives de la lumière réelle du soleil.

- La source prismatique est parallèle au carrelage. Elle éclaire donc la chaise perpendiculairement. Le premier test en tracé de rayon, avant de relancer un rayon secondaire vers la source lumineuse (pour détecter si le point est à l'ombre), est l'étude du signe du produit scalaire du vecteur normal au point d'intersection (un point du carrelage) et du vecteur dirigé vers la source lumineuse. Dans le cas d'un produit scalaire négatif ou nul (ce qui est notre cas), il n'y a pas éclairement du point par la source. En fait, tout l'éclairement du carrelage provient de l'éclairement indirect dû aux particules de poussière englobant notamment la chaise. Une source prismatique arrivant dans une direction non perpendiculaire au carrelage, aurait entraîné une zône d'ombre (la figure 4 en est un exemple).

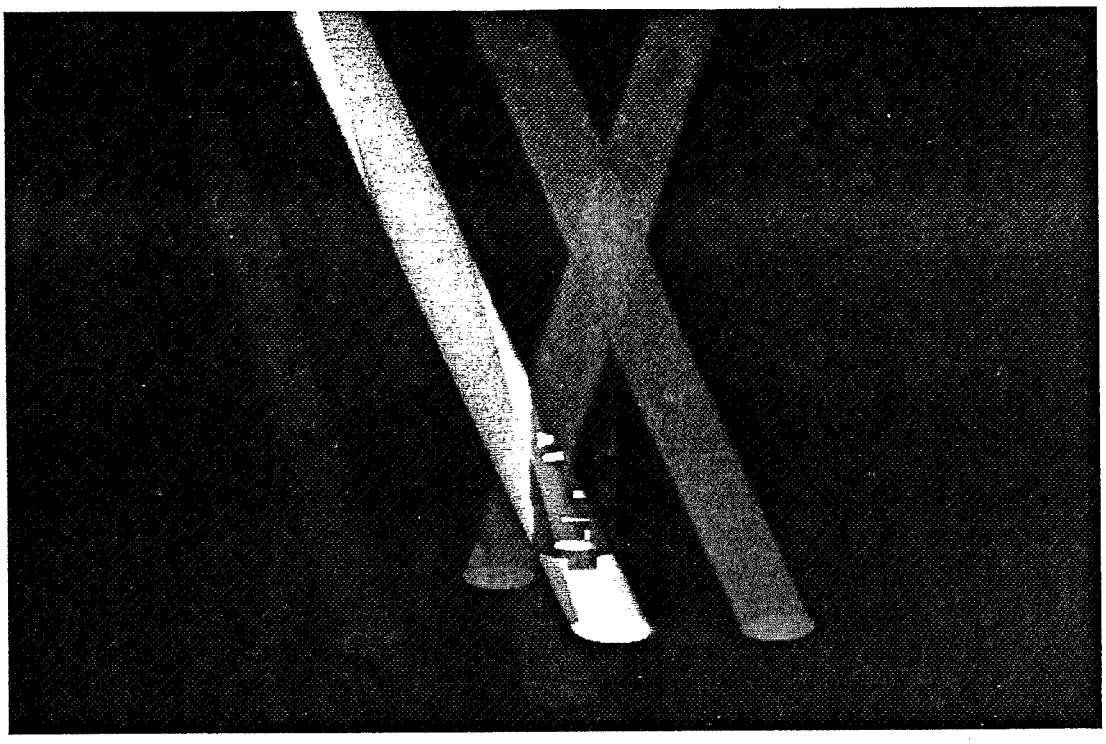


Image 6 : Quelques spots de lumière sont situés devant le cylindre creux. La présence de poussière devant le cylindre se traduit par un apport de lumière, qui diffère selon la couleur des spots.

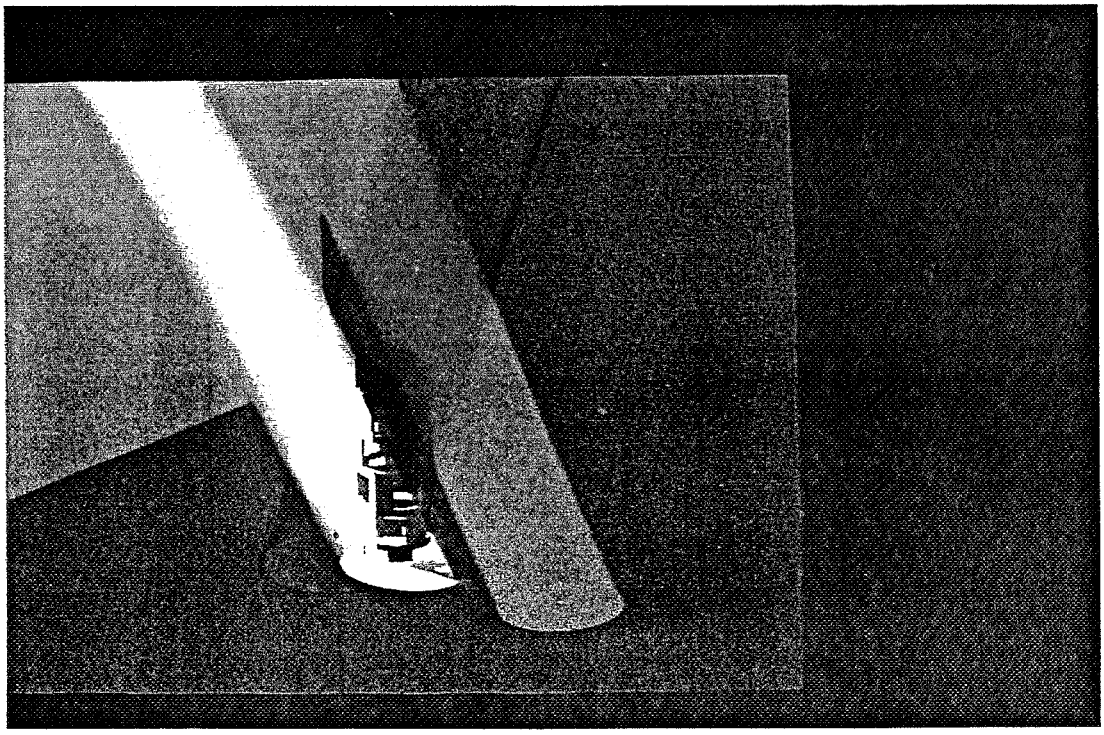


Image 7

Dans cette image, le cylindre est éclairé directement par le spot de couleur blanche qui de ce fait, projette l'ombre du cylindre sur le sol. Les autres spots, rouge et vert, sont situés devant le cylindre et ajoutent alors un éclairement coloré.

Les temps de calculs de ces images sont mentionnés dans le tableau ci-après. Chaque image est de taille 512x640 et a été calculée sur un SPS90.

Numéro d'image	Nombre de primitives	Temps de calculs CPU en h : mn : s
1	3	0 : 41 : 10
2	422	4 : 45 : 04
3	246	3 : 25 : 12
4	19	1 : 41 : 34
5	534	4 : 29 : 03
6	22	2 : 22 : 04
7	20	2 : 03 : 04

6 Conclusion

Bien que nous ayons atteint nos objectifs de départ, plusieurs évolutions sont envisageables. Elles concernent trois directions :

- la géométrie de la source lumineuse. Le fait que les primitives de lumière soient des primitives de l'arbre de construction nous permet d'envisager plusieurs possibilités :

  - l'utilisation d'autres primitives de notre modeleur permettrait d'accroître la diversité des lampes accessibles par l'utilisateur ;

  - la combinaison par le biais des opérations booléennes intersection ou différence (il est important de préciser que toute combinaison de ce type ne se fait qu'avec deux ou plusieurs primitives de lumière et non avec

simples primitives) rendrait possible la création de sources lumineuses à géométrie plus complexe.

Les problèmes d'intersection étant résolus par la méthode choisie, il suffit seulement de décrire la distribution de l'intensité lumineuse associée à la source lumineuse.

Citons quelques idées qui n'ont malheureusement pu être implémentées :

• le premier modèle qui nous soit venu à l'esprit cherche à simuler les lampes demi-sphériques fixées au plafond ou sur les murs. Il serait possible d'envisager de combiner deux types de primitives (voir la figure 10) ;

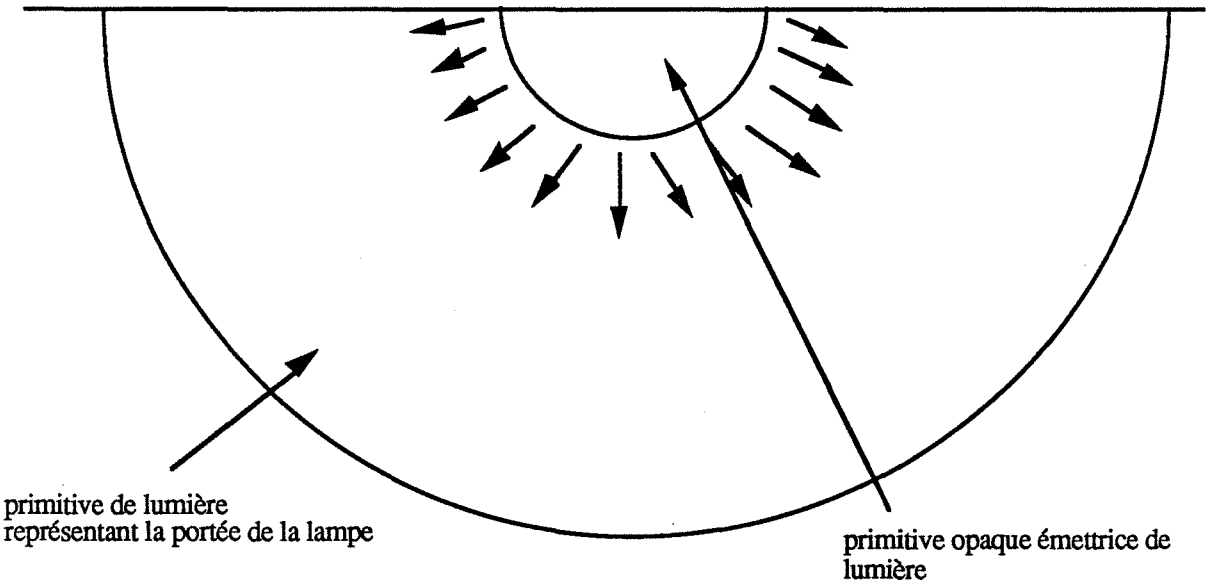


Figure 10 : Combinaison de deux primitives

- la première primitive serait opaque (donc non transparente) et constituerait la zone émettrice de la lampe, au même titre que l'origine du cône ou le disque du cylindre. Elle pourrait être obtenue par l'intersection d'une sphère et d'un cube. L'intensité serait constante en tout point de cette primitive ;

- la seconde primitive représente la lumière émise par la primitive opaque. De même que la première primitive, elle serait obtenue par l'intersection entre un cube et une sphère.

La distribution de l'intensité lumineuse au sein de cette primitive de lumière (au sens où nous l'avons définie dans notre modèle) est calquée sur la symétrie naturelle de la sphère. Cette seconde primitive constituerait la portée de la lampe.

• Basées sur cette décomposition en deux primitives, d'autres lampes nous semblent possibles :

- des primitives opaques comme un demi-cube, un tore, un cylindre couché (pour un néon) semblent susceptibles de simuler des lampes bien connues dans la réalité. Le modèle de lumière attaché à chacune de ces primitives doit être défini pour chaque cas, après étude ;

- un tronc de cône avec comme base émettrice un disque, se rapprocherait plus de la réalité que notre modèle de source conique ponctuelle. La primitive opaque serait constituée du disque (la base émettrice) et la primitive de lumière serait le résultat d'une différence entre un cône et un cube.

Un problème toutefois se pose. Les sources de lumière seront-elles directionnelles. Si oui, comment seront calculées les ombres portées ? Un échantillonnage des points de ces surfaces (on rejoindrait alors les travaux de Brotmann) serait nécessaire. Des rayons secondaires seraient envoyés du point d'intersection aux points échantillonnés. La simplicité de notre méthode en pâtirait. L'intégration de sources non directionnelles entraînerait irrémédiablement (dans le cas de calcul d'ombres portées) un accroîssement des temps de calcul.

- la simulation de la poussière : notre méthode propose un calcul empirique de l'éclairement tenant compte de l'épaisseur de poussière traversée par le rayon œil-pixel pour chacune des primitives de lumière. Des travaux sont en cours de réalisation ([ROEL 90]) pour prendre en compte de façon plus concrète les lois de la physique. La simulation de la poussière est basée sur une modélisation des particules se trouvant dans le faisceau lumineux. Par analogie aux primitives de lumière, ces primitives de particules permettent d'avoir une connaissance exacte et rapide du volume occupé par ces particules (on imagine aisément toute extension du modèle de particules pouvant représenter un type quelconque d'atmosphère : fumée, brouillard,...). Dans le modèle présenté dans ([ROEL 90]), le calcul de l'éclairement du point d'intersection entre le rayon œil-pixel et un objet dépend de :

- l'éclairement émis par les particules de poussière illuminées par la source. Un modèle de particule a été défini ainsi que l'interaction des particules avec la lumière à l'intérieur de la primitive de lumière ;

- l'éclairement de l'objet en ce point, atténué par le passage à travers la primitive de lumière.

- une analyse spectrale. Dans la version actuelle, les objets, les sources de lumière sont caractérisés par leurs trois composantes R, V, B. Des travaux sont en cours de réalisation pour adopter une analyse spectrale, jugée bien plus réaliste (voir le paragraphe 2.3.3 du chapitre premier). La description des sources lumineuses sera donc différente et leur prise en compte, au niveau du calcul d'éclairement et du rendu des couleurs, nécessitera alors certains calculs plus complexes. Les travaux de Roelens cités plus haut, sont les premiers pas dans cette direction.

Chapitre 3

Une Interface Evoluée de CASTOR

1 Introduction

Comme nous l'avons indiqué en introduction de ce manuscrit, la réalisation d'images de synthèse passe nécessairement par la constitution de la base de données graphiques de la scène à visualiser. Cette partie, intitulée saisie de la scène, fait ressortir deux problèmes.

- **le problème de l'interactivité :**

Le choix de l'interactivité se pose et deux attitudes en découlent :

- *l'interactivité est jugée indispensable.* L'utilisateur bénéficie alors d'un outil interactif graphique. Toutes les dernières techniques de l'informatique interactive sont utilisées afin de permettre à l'utilisateur de créer interactivement sa scène devant sa console graphique. A l'aide de menus, des messages, de la souris, il modifie ou construit les entités au fur et à mesure de l'avancée de son travail. Chaque action de l'utilisateur est visualisée et suivie d'un message lui indiquant si l'opération effectuée est correcte et s'il peut poursuivre son travail. Un véritable dialogue est instauré entre l'utilisateur et le programme. En présentant son logiciel qui possède les propriétés ainsi requises, Guinemer ([GUI 89]) affirme la facilité d'apprentissage et d'utilisation de tels logiciels par des non spécialistes. A défaut d'être un spécialiste dans le domaine précis concernant les objets que l'on modélise, le concepteur se doit cependant de posséder des connaissances mathématiques ainsi qu'une certaine familiarité avec la géométrie tridimensionnelle. Nous prétendons donc que ces outils informatiques restent commodes dans une certaine mesure et paraissent vite fastidieux dans le cas de scènes chargées (plusieurs dizaines milliers de polygones par exemple). Autre fait concernant l'interactivité : elle nécessite une visualisation rapide. La visualisation fil-de-fer s'avère donc être indispensable. Elle pourra être suivie bien sûr, d'un autre type de visualisation offrant un meilleur rendu.

- *l'interactivité n'est pas l'objectif recherché.* L'utilisateur construit sa scène à l'aide d'un langage de programmation de description de scènes, lui permettant de définir explicitement les constituants de la scène. CASTOR en est un exemple. Une phase de compilation est nécessaire et interdit donc toute interactivité. Une construction répétitive, des relations entre les objets peuvent être autorisées selon la puissance de ces langages. Ceci pourrait être l'outil absolu si chaque dessinateur était aussi un programmeur.

• **l'aide fournie par le système :**

La constitution de la base de données graphiques de la scène à visualiser est, dans la majorité des cas, lourde et difficile

- au niveau du positionnement des objets :
  Lucas ([LUCA 89]) emploie, à juste titre, le terme de construction *artisanale* pour parler de la difficulté éprouvée pour disposer les éléments d'une scène complexe. Ce problème se pose

  - pour un système de modélisation interactif : comment par exemple, garantir l'alignement en trois dimensions de plusieurs entités ;

  - pour un langage de description : l'utilisateur se doit d'y répondre en donnant les paramètres des transformations. La connaissance de ces paramètres, chose laborieuse, est souvent entachée d'erreurs. La visualisation n'est pas immédiate (une phase de compilation empêche toute interactivité) et nécessite le plus souvent plusieurs essais avant d'arriver à un résultat visuel correct (visuel seulement car la cohérence, comme pour l'outil graphique, n'est pas assurée).

- au niveau de la description de tous les paramètres nécessaires à la visualisation :
  La donnée des coefficients d'éclairement, des coefficients propres aux matériaux, des textures, des différents types d'atmosphères, ... peut paraître rapidement fastidieuse dans le cas de scènes très réalistes. De plus, cela demande à l'utilisateur certaines compétences en colorimétrie ou en photométrie.

Il ressort de ces constatations la nécessité d'une aide du système afin de faciliter la tâche au concepteur par le biais d'outils simples à utiliser.

Au sein de notre système ILLUMINES, la partie saisie offre à l'utilisateur plusieurs modules indépendants entre eux (voir la figure 1). Ils ont pour seul point commun de créer le flot d'entrée à fournir au programme CASTOR, sous la forme d'un fichier.

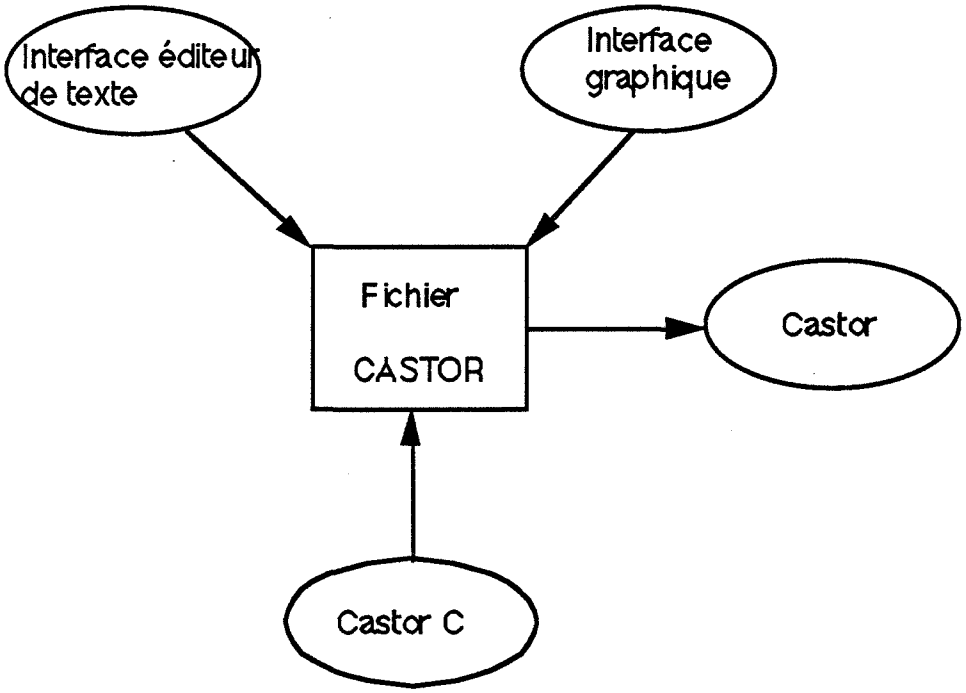


Figure 1 : La partie saisie d'ILLUMINES

La première des interfaces est tout simplement l'éditeur de texte. L'utilisateur crée son propre fichier, écrit dans le langage de description de scène CASTOR, avec l'éditeur de texte "vi", éditeur pleine page d'UNIX. L'apport de filtre (le préprocesseur C) ou d'une sur couche (castorC, [MICH 89]) permet une écriture ainsi qu'une lecture de fichiers *castor* assez fonctionnelles. Le défaut majeur, propre à ce type d'interface concerne, nous l'avons dit, l'interaction, non réalisable.

Le deuxième type d'interfaçage est purement graphique et fait partie des outils graphiques interactifs décrits précédemment. L'utilisateur n'a pas besoin de connaître au préalable la modélisation par arbre de construction et encore moins le langage CASTOR. Tout ceci est effectué dans la plus parfaite transparence. La construction de la scène se fait à l'aide de menus, d'icônes, de multifenêtrage, afin de bâtir l'arbre de construction. Au fur et à mesure de la manipulation, une visualisation en fil-de-fer est effectuée, sans passer par le programme CASTOR. L'interface graphique constitue donc une entité indépendante de

CASTOR, possédant un algorithme de visualisation interne, rapide et simple. En sortie, un fichier CASTOR est automatiquement créé ; tous les rendus disponibles dans ILLUMINES sont alors possibles. Une première interface a été développée et implantée sous Sunview ([AMAR 89]). Une seconde est en cours de développement sous GPR (Domain graphics primitive resource system : produit Apollo) par Philippe Jaillon. La nécessité d'une visualisation sans élimination des parties cachées rend vite ce type d'interface peu agréable dans le cas d'une scène complexe. De plus l'utilisateur manipule uniquement des primitives, la notion d'objet n'existant pas.

Nous disposons donc, au sein d'ILLUMINES, de deux types d'interfaces distinctes regroupées autour de CASTOR. L'aide fournie par le système y est totalement absente. Nous avons donc été amenés à réfléchir sur la conception d'un autre type d'interface graphique plus évolué. Nos objectifs sont les suivants :

- permettre à l'utilisateur de concevoir sa scène de façon naturelle à l'aide de fonctions de haut niveau ;

- offrir une aide du système pour faciliter la conception des scènes, notamment pour le positionnement des objets.

Dans l'optique d'une telle interface, un environnement de base (environnement de programmation, de développement, d'exploitation, ...) s'avère indispensable. Le choix de cet environnement a constitué la première étape de notre étude de faisabilité. Une interface graphique semble obligatoire afin de permettre un dialogue constant durant la création de l'image. Un dialogue avec un utilisateur, ne l'oublions pas, non informaticien et encore moins mathématicien. Les possibilités de fenêtrage, de menus, de graphiques, l'interaction possible avec la souris, le cliquage, sont des atouts importants pour un tel dialogue. Peu de caractères à taper, pas de nombres ou chiffres à donner, tout est dans ou à côté de l'image. L'interface graphique s'avère donc primordiale, dotée, bien sûr, d'un environnement adéquat.

Devant la difficulté du problème, une étude des différents travaux déjà réalisés nous a permis, d'une part de résoudre le problème du choix de l'environnement de programmation, d'autre part d'examiner différentes méthodes mises en place dans les domaines sensiblement voisins que sont la CAO, la robotique, la géométrie, ... Le paragraphe 2 présentera ces différents travaux. Dans le paragraphe 3, nous préciserons les spécificités souhaitées pour notre interface. Le paragraphe suivant sera une description de l'interface évoluée réalisée. Enfin nous terminerons ce chapitre en évoquant les extensions possibles, nombreuses et diverses.

2 Survol des différents travaux existants

Devant l'importante diversité des travaux, nous allons tenter de les classer en fonction des réponses qu'ils apportent aux deux problèmes soulevés dans le paragraphe précédent.

Commençons par les travaux concernant l'interactivité,

• Brock, Coquillart et Cros présentent dans ([BROC 89]) un modeleur géométrique interactif 3D. Une visualisation permanente des objets sous différentes vues est possible. Une zone de dialogue est entretenue avec l'utilisateur par le biais de la souris, du clavier et de menus. L'utilisateur conçoit un objet par l'association de primitives ou de surfaces. Dans le cas des surfaces, deux modes d'actions sont autorisés

- au niveau des points de contrôle ;

- au niveau des tangentes.

• Van Emmerik ([VAN 88]) propose un logiciel interactif de conception de scènes modélisées par un arbre de construction. L'utilisateur travaille interactivement sur un arbre géométrique dont les noeuds représentent des points de contrôle décrivant le positionnement, l'orientation et les dimensions d'une primitive. Toute action (création ou modification) sur ces noeuds est répercutée interactivement, et de façon transparente pour l'utilisateur, sur l'arbre de construction.

• Jense ([JENS 89]) évoque les avantages de la modélisation par les voxels. Ce type de modélisation possède plusieurs inconvénients dont le principal est son encombrement mémoire qui se traduit le plus souvent par un accroissement des temps de calcul. Jense présente différents travaux sur le calcul par machine des transformations géométriques (rotation, translation, affinité) sur les voxels permettant ainsi une manipulation interactive.

Les travaux suivants, très nombreux, traitent de l'assistance du système dans la conception en général (mécanique, robotique, images de synthèse, ...). Evoquons-les en fonction du domaine auquel ils appartiennent.

• la synthèse d'images :

Dans tous les travaux utilisant les principes de l'informatique traditionnelle, l'homme doit se mettre à la portée de la machine. L'utilisateur effectue une description de la scène, composant par composant, à partir d'entités géométriques qu'il positionne tant bien que mal. Sur ce point, on peut noter les travaux de Brock, Coquillart et Cros qui proposent une grille

constituée de lignes horizontales, verticales ou obliques, afin d'apporter une réponse au problème du positionnement des objets. Une autre démarche, plus humaine, a pour principe de mettre la machine à la portée de l'homme. L'utilisateur décrit les objets en citant, à l'aide d'un langage naturel, ses propriétés géométriques, topologiques, mécaniques, ... : un gros cube proche d'un cylindre allongé, ... Cela sous-entend que l'ordinateur possède suffisamment de connaissances dans le domaine concerné, pour pouvoir déduire une description de l'objet ainsi défini. Cette modélisation, dite déclarative, utilise une technique de l'intelligence artificielle : les systèmes experts. Précisons un peu ce nouveau paradigme de programmation ainsi que ses répercussions sur la modélisation déclarative :

> *les systèmes experts* : il semble difficile de donner une définition d'un système expert en quelques lignes. Disons simplement qu'un système expert cherche à simuler une activité intellectuelle. Il est constitué d'une base de connaissances autour de laquelle il va émettre un raisonnement dans une certaine logique. La base de connaissances est constituée de connaissances factuelles et opératoires. Les connaissances factuelles décrivent les faits établis ou à démonter, propres à l'application. Les connaissances opératoires représentent les connaissances obtenues auprès de l'expert exprimées dans le langage de représentation des connaissances, sous la forme de règles. Ainsi la connaissance de l'ordinateur devient essentiellement déclarative : **Si** *condition* **alors** *conclusion*.. Le mécanisme de raisonnement, appelé plus communément le moteur d'inférence, met en relation les connaissances factuelles et opératoires pour produire de nouveaux faits ou règles. Ce mécanisme permet d'établir, à partir de certaines propriétés simples à énoncer (les faits), une description plus importante, plus complexe que n'aurait pu concevoir le créateur. Lucas et al ([LUCA 89]) adoptent une telle modélisation dans le cas du projet ExploFormes offrant la possibilité, entre autres, d'engendrer automatiquement tous les objets d'un univers de formes particulier (exemple : les polyèdres). Les méthodes de l'intelligence artificielle permettent ainsi à l'ordinateur d'explorer l'univers des formes potentielles afin de sélectionner celles correspondant à la définition donnée. L'utilisateur choisit alors la ou les formes qui lui conviennent. A noter aussi que la connaissance des propriétés géométriques, topologiques et mécaniques permet également un calcul de vue automatique susceptible de visualiser les parties intéressantes de l'objet.

Djedi et al ([DJED 89]) proposent un autre type d'aide, toujours dans cette même optique de modélisation. Ils utilisent la description déclarative pour définir les propriétés d'éclairement des objets. Un ensemble de règles est mis en place afin de déduire les différents coefficients de spécularité, de transparence et de réflexion propres à chaque type de matériau mais également de déterminer la meilleure vue tenant compte des éventuels objets transparents.

Intérêt

Cette approche de la modélisation a beaucoup de ressemblance avec celle que nous envisageons. Apporter une aide à la conception de la scène est bien sûr le point commun essentiel entre nos deux approches. Mais aussi la possibilité offerte à l'utilisateur de s'exprimer naturellement durant la phase de conception. Lucas et Djedi emploient la decription déclarative. Nous, nous mettons à la portée de l'utilisateur des fonctions de haut niveau qui font référence à des actes de tous les jours. Leurs travaux sont également en pleine évolution. Nous retiendrons donc de leurs premiers résultats :

- la puissance des méthodes de l'intelligence artificielle permet d'apporter une réponse à ce type d'approche. Les travaux de Plemenos ([PLEM 89]) et de Djedi en sont les premiers fruits.

- la programmation par règles s'impose dans l'optique d'une telle modélisation.

• **la robotique :**

A l'intérieur de ce vaste domaine qu'est la robotique, une technologie bien précise a attiré notre attention : les robots informatisés. Durant la dernière partie de l'histoire de la robotique, l'informatique a été appelée à jouer un rôle significatif en tant qu'outil de programmation et de pilotage. Le désir sans cesse grandissant de munir le robot d'un pouvoir d'*auto-adaptation*, d'un *comportement intelligent*, accentua ce phénomène. Très logiquement, l'intelligence artificielle prit part à cet apport, afin de fournir une véritable *capacité décisionnelle*, un *comportement intelligent de type intellectuel*. Parmi les travaux réalisés, nous nous sommes intéressé à ceux de Laugier ([LAUG 87]) (qui se situe dans le cadre de la réalisation d'un robot intelligent). Ses travaux contribuent à résoudre le problème de l'autonomie de l'action dans le domaine de la robotique d'assemblage. Il s'agit de doter le robot de fonctions logicielles lui permettant d'agir intelligemment sur son environnement, en fonction du contexte d'intervention et du but à atteindre. L'assemblage d'objets nécessite la préhension et le placement des objets. Ainsi, il est demandé au robot

- d'effectuer un mouvement dans la direction de l'objet considéré. La *planification de mouvement* est un problème très important et très complexe mais qui ne nous concerne pas, étant donnée notre application.

- de rechercher la prise ou le positionnement le plus approprié. Le principe consiste à "filtrer" l'ensemble des solutions possibles. Ces solutions sont établies au préalable sur la base d'une analyse des propriétés locales des contacts. Une analyse des propriétés topologiques des contacts permet de déduire des *contacts potentiels* ainsi que des *prises potentielles*. Un

raisonnement, appelé *raisonnement morphologique*, filtre toutes ces solutions potentielles et détermine également les contraintes de mouvement établies par ces contacts. Pratiquement Laugier restreint les possibilités de contacts dans une optique de faisabilité. Tout contact est constitué de faces, de sommets ou d'arêtes. Les surfaces gauches sont écartées. Les contacts sont obtenus par la combinaison d'objets à l'aide des opérations union, intersection et différence. La préhension ou le positionnement d'un objet par un robot exige à partir de la connaissance exacte des contacts, une solution exacte. Laugier utilise à ce niveau la puissance de l'intelligence artificielle. Il montre ainsi que la réalisation d'un système de programmation automatique de robots n'est pas une utopie. L'élargissement des tâches à effectuer, la suppression des restrictions sur les contacts ou sur le processus de planification (aucune rotation n'est possible), le font cependant préconiser une démarche semi-automatique. Le système doit être capable de proposer interactivement des solutions. L'opérateur pour sa part valide les choix proposés et guide le système en cas d'échec.

Intérêt

Cette deuxième phase de traitement a beaucoup de points communs avec notre application. Notre objectif, infiniment plus modeste dans l'immédiat, a pour but de proposer une solution intermédiaire. L'utilisateur, par le biais de fonctions de haut niveau, aura la possibilité de placer son objet comme il convient. Dans l'optique d'une étude de faisabilité, nous retiendrons des recherches dans le domaine de la robotique, plusieurs aspects importants :

- une restriction des contacts est nécessaire ;

- une démarche semi-automatique s'impose ;

- l'apport de l'intelligence artificielle apparaît inévitable.

• la géométrie bidimensionnelle et l'Enseignement Assisté par Ordinateur :

La plupart des travaux auxquels nous nous sommes intéressés se rapprochent de notre application au niveau de l'objectif visé : assister l'utilisateur en lui offrant des outils simples à utiliser. Dans une optique d'enseignement, la construction géométrique s'effectue effectivement en précisant de façon naturelle les qualités ou propriétés des courbes ou traits que l'utilisateur désire. La médiane, la tangente, la bissectrice, un triangle isocèle peuvent se définir de façon déclarative : "la médiane dans le cas d'un triangle, est un segment de droite joignant un sommet au milieu du coté opposé". Citons quelques exemples parmi les nombreux travaux réalisés

- Chouraqui et Inghilterra ([CHOU 87]) ont réalisé un système expert d'enseignement assisté par ordinateur. Leur méthodologie adopte une hiérarchie orientée objet pour les objets géométriques et les définitions mathématiques des figures se traduisent sous forme de règles : "théorème = règles". Ils utilisent pour ce faire un langage orienté objet, OBJ-LOG, hautement interactif, développé autour de Prolog.

- Braun ([BRAUN 89]) présente un outil expérimental permettant une modélisation aisée, en 2D, de connaissances géométriques grâce à un formalisme orienté objet spécialement conçu à cet effet : Géophile. Un concept nouveau intervient : la notion de contrainte. La contrainte d'un trait ou d'une autre figure réside dans sa propriété : la médiane d'un segment passe par le milieu du segment. La contrainte "passe par le milieu" est instaurée et devra bien sûr être préservée. La complexité des figures peut entraîner plusieurs contraintes pour une seule figure ; toute modification de cette figure (par l'intermédiaire de l'un de ses composants : un des points du segment par exemple) nécessitera une modification afin de préserver la ou les contraintes à son égard. Le terme de programmation par les contraintes est d'ailleurs souvent employé, à juste titre ([BORN 89]). Braun utilise toute la puissance de l'environnement orienté objet (à base de LeLisp) pour permettre la définition des constructions géométriques de façon incrémentale ou déductible et pour assurer la gestion automatique des contraintes en cas de modifications. Braun envisage de faire, à court terme, de Géophile un progiciel destiné à l'enseignement.

- Schonek et Verroust ([SCHO 89]) proposent également une construction par les contraintes. S'inspirant de ([SUND 87]), ils utilisent deux contraintes de base : CA ou contrainte d'angle entre deux segments ; CD ou contrainte de distance entre deux points. Toute autre contrainte est ensuite transformée en un ensemble de CA et/ou de CD. Contrairement à Braun, ils utilisent la programmation par règles. Différents types de règles existent :

- les règles de création : elles permettent d'établir les CA et les CD ;

- les règles de construction : ces règles permettent au fur et à mesure d'une construction géométrique (un triangle par exemple), de regrouper différents ensembles de CA ou de CD en un seul ensemble (le triangle est un seul ensemble de CA ou de CD). Citons l'exemple de la règle qui regroupe deux ensembles de CA qui contiennent en commun un segment. Schonek et Verroust proposent, basées sur ce principe, des règles de construction de triangles, de quadrilatères et de parallélogrammes.

- les règles de vérification de contraintes : elles déterminent si la modification de la figure considérée (un ensemble de CA ou de CD) est possible.

Intérêt

Nous retiendrons de ces travaux :

- la notion de contrainte : elle nous semble importante dans la mesure où certaines des fonctions de haut niveau envisagées (poser sur, coller contre,...) cherchent en fait à établir une contrainte entre les objets considérés. De la détermination d'une droite en précisant qu'elle passe par le milieu de deux points, à la détermination de la position d'un cube dans l'espace en le collant contre un autre cube, le principe de construction est le même. Le positionnement des objets se fait par rapport aux autres et en définissant les propriétés géométriques souhaitées. L'aide du système intervient dans l'analyse, la compréhension et la traduction de ces propriétés. La puissance de raisonnement du système et la richesse des fonctions de contraintes sont primordiaux pour permettre une conception par le biais des fonctions de contrainte.

- la conservation des contraintes : le deuxième objectif que nous nous sommes fixés (aide à la modification des objets) consiste en fait à préserver les contraintes mises en place lors du positionnement. Offrir de façon transparente une telle aide est possible ; Schonek et Braun en apportent la preuve.

- la programmation par règles : l'approche de Schonek a retenu notre attention. La puissance de la programmation par règles ressort ici face à l'éventuelle combinatoire du problème. L'élaboration d'un tel logiciel avec une distinction nette entre règles de construction et règles de conservation, laisse entrevoir une plus grande souplesse.

• La Conception Assistée par Ordinateur

Le domaine de la CAO (ou CFAO) (mécanique, architecture, conception automobile, ...) comporte plus que de simples ressemblances avec celui de la création d'images de synthèse tel que nous le voyons. Plusieurs points les rassemblent :

- Une aide du système est attendue afin de permettre un assemblage aisé des composants et ceci sans risque d'erreur d'appréciation (deux composants collés le sont de manière exacte, aux erreurs d'imprécision numérique près) ;

- Les techniques de modélisation sont le plus souvent les mêmes (CSG, BREP). Hanser ([HANS 90]) évoque les difficultés rencontrées pour la génération de contraintes complexes (tangentes, ...) entre les surfaces et conclut en signalant que seuls sont au point, les systèmes basés sur une

modélisation du type CSG. Schonek ([SCHO 89] est également très pessimiste quant à l'extension 3D de sa méthode de construction géométrique ;

- Une optique d'interactivité est présente dans la majeure partie des systèmes de CAO.

Beaucoup de points communs existent donc. Les effets de rendu sont cependant souvent moins performants que ceux utilisés en synthèse d'image (ombres, transparences, ...). Dernière différence que nous citerons et qui réside dans le but recherché de la création : l'image de synthèse n'a pas toujours la prétention de présenter des objets vérifiant certaines lois physiques, mécaniques.

Notre recherche se devait donc de porter un regard sur ce domaine, en constante évolution depuis sa naissance. Sans aucune prétention de notre part, nous allons essayer de suivre son évolution, depuis les premiers systèmes de CAO classiques jusqu'aux systèmes intelligents actuels. La littérature abonde. Nous n'en avons considéré qu'une petite partie nous permettant quand même de pouvoir dégager les choix effectués tout au long de l'histoire de la CAO.

De la CAO basée sur la géométrie à la CAO basée sur l'intelligence artificielle

Le processus de conception d'un produit industriel est constitué de plusieurs étapes (conception géométrique, simulation du comportement, optimisations, ...). Les premiers systèmes de CAO intervenaient uniquement au niveau de la phase de conception géométrique. L'aide procurée par l'ordinateur à l'infographiste consistait surtout à fournir des outils d'édition sophistiqués et d'affichage rapide. Il en résultait, durant la conception, un décalage entre la représentation géométrique de l'objet affiché et la représentation mentale que s'en faisaitt le créateur. Hanser ([HANS 90]) définit négativement ces systèmes classiques de systèmes "calculette" et affirme que le "A" (de conception assistée) n'est pas justifié. Rapidement, les systèmes permettent à l'utilisateur de définir explicitement des contraintes éventuelles ou des relations entre objets. Pratiquement cette aide se matérialise par l'existence de points, d'axes caractéristiques que l'utilisateur a tout le loisir de désigner afin de manipuler les objets ainsi concernés. Aligner deux axes, faire coïncider deux points revient à établir une contrainte géométrique entre les deux objets (dans un système de CAO traditionnel, il n'existe que des contraintes géométriques). Toute contrainte ainsi établie est traduite par une ou plusieurs équations. Le problème se ramène à la résolution d'un système (non linéaire) d'équations où les inconnues sont les coordonnées des points caractéristiques. Dans une optique d'optimisation, des méthodes de calculs sont proposées ([LIN 81]). Schonek ([SCHO 89]) en présente un survol. Citons Lee ([LEE 85a]) qui propose une structure hiérarchique dans laquelle les composants d'un bloc sont reliés entre eux par ce qu'il appelle un "lien virtuel". Véritable structure de données, un lien virtuel contient aussi bien des informations géométriques que des informations sur la nature de la liaison (attachement rigide ou pas, contrainte de translation, ...) ne nécessitant pas ainsi de connaître

les matrices de transformations pour chacun des composants ([LEE 85b]). Basés sur cette structure, il propose par la suite différents systèmes de modélisation pour l'assemblage ([ROCH 87], [KIM 89]). Différentes fonctions d'assemblage sont alors accessibles : aligner deux axes caractéristiques, coller deux faces caractéristiques. Parallèlement, la programmation orientée objet s'affirme et parvient à s'imposer pour toutes sortes de modélisations ([GRAN 86], [SABE 87], [FLEI 87], [BEEK 89], [SMIT 89], [LAUN 89]). La puissance du modèle objet permet de faire une meilleure conceptualisation et abstraction des problèmes : quand on pense à quelque chose, on y pense en termes de classes d'objets et de comportements. De nombreux systèmes ont été réalisés (les produits de la société Intergraph, Euclid-Is de la société Matra Datavision pour ne citer qu'eux) et ouvrent la voie à l'intelligence artificielle. La conception des systèmes offrant une CAO encore plus intelligente est effectivement demandée. Un raisonnement, un contrôle intelligent, des objets intelligents, la notion de contraintes autres que géométrique sont alors rendus possibles par les méthodes de l'intelligence artificielle. Arbab [ARBA 89]) précise même que le formalisme de la description déclarative évite la propagation des erreurs ou imprécisions numériques. De nombreux systèmes intelligents existent ([GERO 87], (OHSU 89], [SMIT 89]). Pratiquement, Oshuga le précise, les applications de l'intelligence artificielle concernent des modules restreints et séparés qui doivent s'interconnecter à un système de CAO classique. Citons les travaux ([KIMU 87], [BERL 88], [SHEU 88], [DAHS 89], [KRAU 89], [CHOU 90], [MACU 90]), où l'apport de l'intelligence artificielle permet d'obtenir une conception plus rapide, plus facile et plus sûre. La construction par contraintes est souvent utilisée ([BERL 88], [DAHS 89], [MACU 90]). Une réaction de la part du système ([KIMU 87], [SHEU 88]) est proposée par la propagation des contraintes. Prenons le cas de Scheu qui autorise une modélisation interactive (à l'aide de la souris, de boutons,...) du type CSG. Les éléments de l'arbre CSG ainsi que les relations les liant, sont formulés de façon logique. Scheu met à la disposition de l'utilisateur les fonctions "placer contre", "bouger" avec répercussion du mouvement sur les objets posés dessus celui qui est manipulé. Une base de connaissance orientée objet est utilisée. Tout évênement associé à un objet (tout est objet, en particulier les éléments de l'arbre CSG) est traduit, de façon formelle, dans une transaction déclarative.

L'apport de l'intelligence artificielle aux systèmes de CAO est encore de nos jours en pleine expansion. Oshuga dégage les points essentiels, selon lui, pour la réalisation de systèmes de CAO intelligents :

- une connaissance déclarative est nécessaire pour les raisons suivantes :
  - une telle description est très lisible car proche du langage naturel ;
  - elle conserve la cohérence de la connaissance durant les étapes de la conception ;
  - elle permet d'émettre un raisonnement, un contrôle intelligent.

- un environnement orienté objet est nécessaire. Il affirme en effet que tout logiciel d'intelligence artificielle qui se veut efficace et puissant doit allier programmation orientée objet et programmation logique.

- une programmation procédurale doit être possible. Les méthodes de l'intelligence artificielle émettent des raisonnements dont la finalité peut consister à modifier la position d'un composant. Un calcul numérique est à effectuer. La programmation traditionnelle reste encore, à ses yeux, indispensable pour de telles opérations.

Intérêt

Nous retenons de tout ce vaste domaine plusieurs constatations :

- Afin de prétendre à des systèmes performants, l'intelligence artificielle est apparue comme primordiale. Elle peut émettre un raisonnement dans le but de répercuter une action ([SHEU 88]) ou en déduire les différents modèles géométriques possibles ([KRAU 89]) ;

- un environnement orienté objet est nécessaire ;

- la construction par les contraintes est de plus en plus utilisée. Naturelle, elle permet également une conception rapide et cohérente.

CONCLUSIONS SUR LE SURVOL

Nous n'allons pas énumérer de nouveau toutes les idées ou constatations qui émanent de ce survol. Trois points essentiels méritent cependant d'être accentués tant ils ressortent de l'ensemble des travaux :

• un environnement orienté objet est primordial dans l'optique de l'interactivité ;

• les méthodes de l'intelligence artificielle, et plus particulièrement la programmation par les règles, sont nécessaires pour fournir un raisonnement logique ;

• la notion de contrainte, qui est naturelle, est importante dans la mesure où elle ouvre la voie à deux possibilités :

- la construction par les contraintes, qui assure une cohérence du modèle ;

- la propagation des contraintes, qui apporte une aide considérable par la répercussion des modifications sur les objets liés par la contrainte.

Le survol des différents travaux a également été bénéfique pour deux autres raisons :

• Il fut déterminant dans le choix du langage de représentation des connaissances. Parmi les langages disponibles sur le marché, KOOL[1] (Knowledge representation Oriented Object Language) est le langage qui a retenu notre attention. Deux aspects importants, fortement influencés par les remarques précédentes, guidèrent notre choix :

- Kool réunit les trois paradigmes de programmation : Kool est un environnement de développement de systèmes experts construit autour d'un langage objet, sous un environnement LeLisp. Kool est un générateur de systèmes experts qui combine donc un langage objet et un système de règles doté d'un moteur d'inférence. Une caractéristique de Kool est son uniformité : tout est objet, y compris les règles. Il s'ensuit un moteur d'inférence assez souple et puissant.

- Kool est un produit fini, commercialisé. Effectivement, Kool c'est aussi un environnement de développement (environnement de programmation graphique, outil de définition d'interfaces de dialogue personnalisées, ...) et tout un environnement d'exploitation.

• Ce survol nous a permis de mesurer nos objectifs. Ils ne nous semblent pas utopiques. Tous ces travaux, sensiblement similaires, en sont la preuve.

3 Spécifications de notre interface évoluée

Notre interface évoluée devra satisfaire les principes des outils graphiques mentionnés au paragraphe 1, à savoir :

• l'interactivité ;

• la transparence et l'indépendance vis-à-vis de CASTOR : l'utilisation d'un langage peut s'avérer réaliste à condition de posséder un langage interprété et non plus compilé. Djedi et al ([DJED 89]) le confirment dans leur étude sur une modélisation d'images de synthèse qui les feront pencher pour un autre type d'approche que nous détaillerons ci-après. Le côté interactif souhaité pour notre interface empêche l'utilisation du langage CASTOR. Notre interface devra donc constituer une entité indépendante de CASTOR, avec une

[1] Kool est un produit BULL

visualisation en fil-de-fer. La création d'un fichier CASTOR permettra par la suite de bénéficier de toutes les possibilités de rendu du système ILLUMINES.

De plus, notre interface devra offrir à l'utilisateur une aide afin qu'il puisse concevoir sa scène de façon naturelle :

• cette aide devra pouvoir intervenir :

- au niveau du positionnement des objets :

Le positionnement des objets s'effectuera en référençant l'un des objets par rapport aux autres, par le biais de fonctions de haut niveau qui font référence à des actes ayant un sens intuitif très fort pour des non informaticiens ou des non mathématiciens : poser_sur, coller_contre, approcher_de, éloigner_de, ..., tout en prenant en compte les différents cas possibles. Prenons le cas de poser_sur. Le système doit être capable de proposer une solution quelle que soit la forme et la position de l'objet receveur ;

- lors de la modification de la scène :

Le coté imprévu et interactif de la création de la scène (notre interface est destinée entre autres, à des étudiants d'arts plastiques, d'écoles d'architecture, désireux de créer des images de synthèse non forcément réalistes) doit permettre à tout moment, une modification de l'un ou de plusieurs constituants de la scène. Déplacer une table nécessite de déplacer tout ce qui est posé dessus ; grossir un objet fait référence à un point à partir duquel se fera l'homothétie, ... L'aide que nous envisageons de fournir permettra d'effectuer automatiquement ces actes, dans le but de faciliter la tâche à l'utilisateur pour qu'il ne se consacre qu'à la création.

Prenons l'exemple de la figure 2

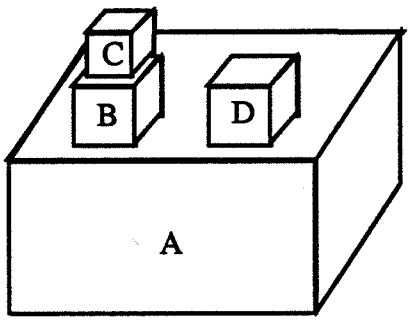


Figure 2 : Un exemple de scène

bouger A nécessite de :	grossir A impose :
translater A	d'effectuer une afinité sur A
translater D et B	de translater D,B (puis C)
translater C	afin qu'ils restent posés sur A.

Avec les interfaces disponibles dans ILLUMINES, toutes ces actions sont à effectuer :

- en modifiant directement le fichier CASTOR par l'intermédiaire de l'éditeur pleine page d'UNIX, "vi". Reste à connaître la modification exacte à effectuer. Une compilation sera ensuite nécessaire pour traiter le nouveau fichier.

- en modifiant interactivement tous les objets ainsi pris en compte, à l'aide des interfaces graphiques interactives. Mais la cohérence globale de la scène n'est pas assurée du tout.

Avec l'interface que nous souhaitons développer, le système génèrerait automatiquement toutes ces actions, avec garantie du maintien de la cohérence de la scène.

• nous souhaitons également faire intervenir un minimum de logique physique au sein de notre système. Par exemple, nous voudrions avoir des réactions du système dans les situations suivantes :

- poser un objet trop lourd doit être signalé comme absurde ;

- poser un objet lourd sur un objet non rigide doit entraîner une déformation de ce dernier ;

- poser un objet sur un plan incliné peut entraîner la glissade de l'objet.

La résolution de ces problèmes nécessite de tenir compte de lois physiques. La résistance à l'écrasement, les lois de déformation élastique ainsi que les coefficients et les lois de frottement sont des phénomènes connus. Notre étude envisagera la possibilité de les implémenter au sein de notre interface.

4 Réalisation ([FERT 89b], [FERT 90])

4.1 Présentation générale

Notre interface est un véritable outil graphique interactif. La construction de la scène se fait objet par objet ; par défaut ceux-ci sont positionnés sur un plan initial de base. A tout moment, l'utilisateur a la possibilité de visualiser son objet, ou sa scène. L'utilisateur peut aussi préciser, pour chaque objet visualisé, le niveau de détail. Une silhouette de l'objet (réduite pour l'instant à la boîte englobante de l'objet) permet d'apprécier sa situation sans trop surcharger l'écran ; les autres objets, jugés plus importants, sont visualisés suivant leur niveau de détail.

Les différentes possibilités de visualisation sont classiques :

- vue de dessus ;

- vue de face ;

- vue de coté.

Un calcul automatique du point de vue est effectué de façon à voir toute la scène, pour des angles de vision donnés similaires aux angles de vision humaine. A partir de ces vues, l'utilisateur peut tourner autour de la scène, s'en approcher ou s'en éloigner .

L'utilisateur a la possibilité de visualiser un objet, deux objets, ..., ou tous les objets de la scène.

L'utilisateur se voit offrir une possibilité purement informatique : la duplication. Une scène peut comporter plusieurs objets identiques (des verres par exemple). Pouvoir dupliquer évite alors d'avoir à modéliser plusieurs fois un même objet.

Durant la conception de la scène, l'utilisateur dispose des primitives du modeleur ainsi que de fonctions de haut niveau.

Ces fonctions de haut niveau s'appliquent

- sur les primitives afin de créer un objet ;

- sur les objets ; l'utilisateur peut concevoir autant d'objets qu'il le désire et les positionner de la même façon que les primitives : par les fonctions de haut niveau.

Les fonctions de haut niveau font référence à des actes de tous les jours et offrent ainsi un moyen de positionnement naturel. Le système se charge, au regard de la situation, de proposer une solution.

Nous distinguons deux types de fonctions :

- *les fonctions de contraintes* :

Ces fonctions ont pour but d'établir des relations entre les objets concernés, relations qui deviennent par la suite des **contraintes**. Poser un objet A sur un objet B revient à amener au **contact** la partie inférieure de A avec la partie supérieure de B. Tout mouvement ultérieur de A sera **contraint** de façon à ce que A reste au contact de B, tout mouvement de B **contraindra** A à effectuer le même mouvement : A et B sont liés par une **contrainte.**

L'emploi du terme **contact** apparaît beaucoup dans les travaux similaires de la robotique où l'assemblage d'objets consiste à positionner les objets à l'aide des contacts. Notre démarche est la même. Etablir une relation revient à faire coïncider les contacts des objets considérés. Ces contacts sont bien sûr propres à la fonction : dans le cas de poser, on définira un contact_haut et un contact_bas alors que dans le cas de coller on utilisera un contact_contre.

L'unique fonction de contrainte que nous avons implémentée pour l'instant est la fonction "poser_sur", que nous présenterons dans le paragraphe 3.4.3.

- *les autres fonctions : grossir, diminuer, éloigner de, rapprocher de, déplacer vers*

Ces fonctions, que nous détaillerons dans le paragraphe 3.4.2, interviennent pour modifier la position ou la forme des objets. Elles ne créent pas de relations entre les objets mais peuvent par contre perturber des contraintes établies par une fonction de contrainte.

Le système intervient de différentes façons, toujours dans l'optique de faciliter la tâche de conception :

- *il fournit des renseignements* concernant les effets éventuels de la modification d'un objet. Ceci est réalisé en offrant à l'utilisateur les informations suivantes:

  • en cas de grossissement ou de déplacement d'un objet, le système signale si celui-ci (ou ceux situés sur lui) arrive au contact d'un autre objet ;

• à la suite du déplacement d'un objet A ou de la pose d'un objet A sur un objet B, le système signale si l'objet A est en déséquilibre ou non.

L'aide du système se matérialise par l'affichage d'un message : "Attention l'objet A va tomber", "Attention l'objet A va toucher l'objet C", ... Si ce message indique à l'utilisateur un effet non recherché, celui-ci a la possibilité d'annuler la modification envisagée. Les calculs sont effectués en travaillant uniquement sur les boîtes englobantes des objets. Cette aide, en plus de sa fonction première, peut amener l'utilisateur à la considérer comme un outil de positionnement : placer l'objet A le plus près possible de l'objet B, par exemple.

- *il évite certains calculs* : modifier la taille d'un objet nécessite de savoir par rapport à quelle origine on effectue l'affinité. Le choix, par défaut, que nous proposons est le suivant : si un objet est posé sur un autre, il doit modifier sa taille de façon à toujours rester posé sur l'autre ; si un objet est isolé, on va modifier sa taille en laissant invariant son centre de gravité.

- *il génère la répercussion des actions* sur les objets liés à l'objet manipulé. Déplacer une table implique de déplacer par exemple tous les verres posés dessus. L'utilisation des contraintes permet de résoudre ce problème. Nous le détaillerons ultérieurement.

4.2 La hiérarchie objet

Nous avons expliqué le principe de construction de la scène, à savoir objet par objet. Une modélisation par arbre de construction correspond assez bien à cette approche, dans la mesure où la conception d'un objet se fait par assemblage de primitives disponibles dans le modeleur.

Nous avons été amené à définir une hiérarchie constituée de la super-classe standard *object* et de trois sous hiérarchies (voir la figure 3).

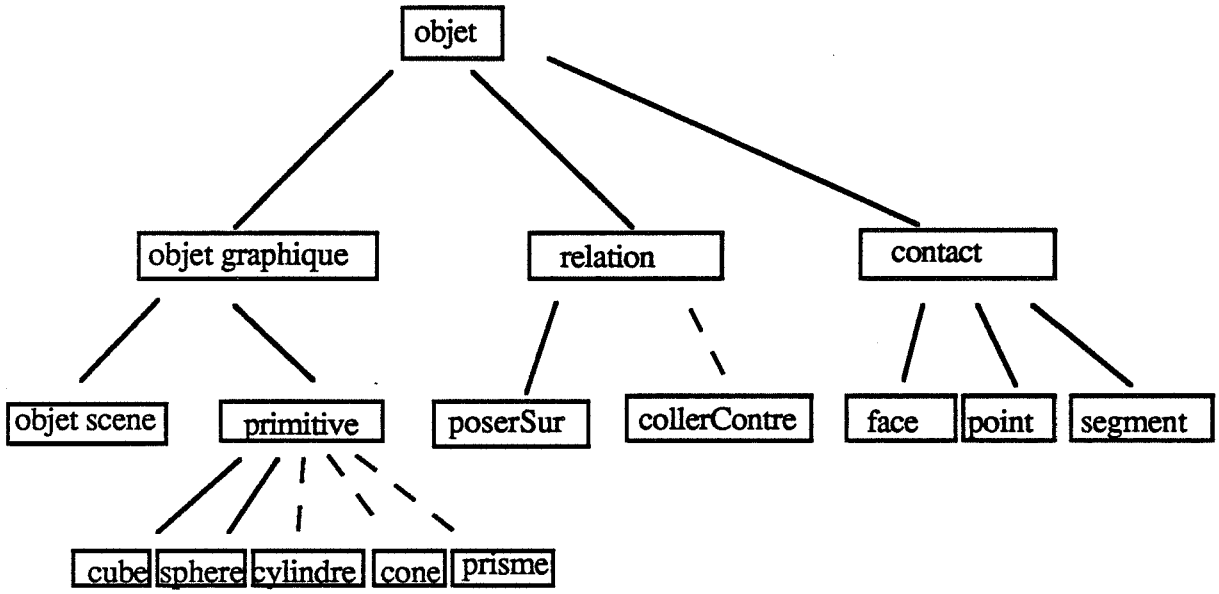


Figure 3 : La hiérarchie objet

4.2.1 la hiérarchie *objet-graphique*

Cette hiérarchie permet de définir les objets de la scène. Elle est constituée de différentes sous classes.

4.2.1.1 La classe *objet_scène* (voir la figure 4)

```
{InstClass : SceObj
Super : objGraph
<-- Poids : ()
        Mode : Mono
        Type : Number
        Init : (set 'p 0)
        WhenFilled:[calPoidSceObj]
    CentreGravite : ()
        Mode : Mono
        Type : PointG
        WhenFilled:[calCentreSceObj]
    Duplicata : ()
        Mode : Mono
        Type : SceObj
    OpBool : ()
        Mode : Mono
        Type : [Union, Feuille]
    Pere : ()
        Mode : Mono
        Type : SceObj
    Fils : ()
        Mode : Multi
        Type : SceObj
        MaxCard : 2
        WhenCard : ajout
        WhenRemoved: [retrait]
    contactHaut : ()
        Mode : Multi
        Type : Primitive
    contactBas: ()
        Mode : Multi
        Type : Primitive
    contrainte : ()
        Mode : Mono
        Type : Relation
    BoiteEnglo : ()
        Mode : Mono
        Type : SphereEnglobante
        WhenFilled:[calBoiteSce]
Methods :
modifier_taille : gros_obj
deplacer : deplacer_obj>
}
```

Figure 4 : La classe *objet scène*

Les instances de la classe *objet scène* représentent les objets composites, à savoir un assemblage de primitives obtenu à partir d'opérations booléennes. Dans un premier temps, toutefois, seule l'union est implémentée en raison de la difficulté de distinguer les intersections ou les différences d'objets visualisées en fil-de-fer. La discussion, présentée au paragraphe 4.2.3, justifiera ce choix.

Tout objet de cette classe possède les attributs suivants :

- *poids* : il permet de déterminer si un objet n'est pas trop lourd pour être posé sur un autre. Nous présenterons un choix purement arbitraire permettant de traiter ce problème. Notre travail, rappelons le, est une étude de la faisabilité d'une interface capable de prendre en compte certains phénomènes, et de réagir en conséquence. Dans le cas de la prise en compte du poids pour la pose d'un objet, nous envisageons, par la suite, de donner une véritable signification au pourquoi de ces comparaisons et de proposer alors certains choix réfléchis. Nous estimons en effet, qu'il est nécessaire dans bien de domaines, d'introduire une règle (en architecture, par exemple, où le poids du toit et celui de la charpente sont très liés).

- *centre de gravité* : il permet de déterminer l'équilibre de l'objet ; il servira aussi à positionner, par défaut, certains objets (voir la fonction poser_sur) ;

- *boîte englobante* : elle intervient

  - pour les calculs d'intersection avec les autres objets ; la boîte englobante d'un objet complexe est une sphère afin de simplifier les calculs.

  - pour le calcul de la vue standard. Une boîte englobante de la scène est déterminée. La position par défaut de l'œil sera telle que toute la scène sera visualisée à l'écran. Bien sûr, en cas de modification des objets, la position de l'œil doit évoluer, d'où la nécessité de cet attribut.

- *contact_haut, contact_bas* : ils représentent les parties des objets considérés que devra prendre en compte la fonction poser_sur : poser A sur B reviendra à poser le contact_bas de A sur le contact_haut de B. Ces attributs diffèrent selon la classe des instances auxquelles ils appartiennent. Dans le cas d'un *objet scène*, contact_haut (respectivement contact_bas) est un ensemble de primitives ;

- *fils* : les instances sont susceptibles d'être composées de plusieurs primitives. Toutes ces primitives se regroupent sous forme d'un arbre binaire. *Fils* contient donc au plus deux objets qui peuvent être des primitives ou des *objets*

scènes dans le cas d'un objet composite comportant au moins trois primitives ;

- *opération booléenne* : dans le cas où l'*objet scène* est constitué d'une seule primitive, cet attribut est vide, sinon il correspond à l'opération booléenne utilisée, à savoir l'union (puisque c'est la seule permise). Cet attribut, associé au précédent, permettra de construire le fichier CASTOR ;

- *contrainte* : il contient la liste des contraintes entre l'*objet scène* et les autres *objets scènes*. Ces contraintes sont des instances de la classe *Relation* ;

- *Duplicata* : offrir à l'utilisateur la possibilité de revenir sur sa décision nécessite de travailler avec des copies des objets. *Duplicata* est donc une copie de *objet scène*.

4.2.1.2 La classe *primitive*

Les instances de la classe *primitive* possèdent les attributs suivants :

- *poids*, *centre de gravité*, *boîte englobante*, *duplicata* : ils ont le même sens que ceux définis ci-dessus ;

- *père* : représente l'*objet scène* auquel est attaché la primitive ;

- *contrainte* : il contient également la liste des instances de la classe *Relation*, mais cette fois-ci entre *primitives* ;

- *Matrice* ;

- *CouleurR*, *CouleurV*, *CouleurB* ;

- *hauteur* : une primitive unitaire a une hauteur égale à 1. Après l'avoir posée sur un autre objet, la hauteur devient égale à 1 plus la hauteur de l'objet récepteur.

4.2.1.2.1 Les classes cube, sphère, cylindre

Les classes *cube*, *sphère*, *cylindre* sont des sous-classes de la classe *primitive*. Elles possèdent par héritage tous les attributs définis au niveau de la classe *primitive*. Cependant, quelques attributs leur sont propres :

- *contact_haut, contact_bas* : dans le cas du *cube*, ces attributs sont des faces, tandis que pour la *sphère*, il s'agit de points et pour le cylindre des faces ou des segments (suivant que le cylindre est debout ou couché).

- *matériau* : nous possèdons au sein du système ILLUMINES différentes fonctions de textures parmi lesquelles les textures de bois, de fer et de plastique. Nous offrons donc à l'utilisateur la possibilité de désigner le type de matériau qu'il désire. Le système plaquera alors la fonction associée et donnera également un poids approprié au type de matériau (une sphère en bois pèse moins lourd qu'une sphère de même taille, en fer). Une fois encore, cette possibilité n'est qu'une étude de faisabilité. D'autres, plus complexes, seraient alors à envisager. Par exemple, donner la possibilité à l'utilisateur d'indiquer des caractéristiques d'éclairement (transparence, spécularité, ..., les travaux de Djedi et al seront alors d'un intérêt certain).

4.2.2 la hiérarchie relation

Les relations à l'intérieur d'un objet ou entre objets sont définies par l'intermédiaire de classes de relations. Tout objet graphique (primitive ou objet scène) possède, nous l'avons vu, un attribut, *contrainte*, qui est la liste des relations le liant aux autres objets. Diverses sous-classes de relations permettent alors de représenter toutes les contraintes possibles. Nous utiliserons uniquement dans un premier temps, la relation *poserSur*, précisant le fait qu'un objet graphique est sur un autre objet graphique. Les fonctions poséContre, colléSur, ... seront envisagées (voir le dernier paragraphe à ce sujet).

Une instance de classe *Relation* possède deux attributs de type *objet graphique* désignant les objets pris en compte par cette relation. Les instances de sous-classes *poserSur* possèdent également d'autres attributs que nous détaillerons dans le paragraphe 4.4.3. Nous insisterons au cours de ce paragraphe, sur l'association entre les instances *Relation* et les fonctions de contrainte. Ces dernières consistent à établir une contrainte qui se traduira par la création de l'instance *Relation* associée. Il se déclenchera alors une réaction du système (voir le paragraphe 4.4.3.2). La construction par contrainte mérite alors pleinement son nom ([FERT 90]).

4.2.3 la hiérarchie contact

Les diverses fonctions de manipulation (poser, ...) mettent en jeu les contacts respectifs des objets considérés. La définition des contacts s'avère donc importante et a fait l'objet de beaucoup d'études, dans le domaine de la Robotique notamment.

En ce qui nous concerne, nous avons analysé plus particulièrement les types de contacts suivants :

• **les contacts inclinés** (voir la figure 5)

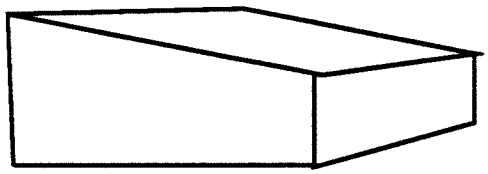


Figure 5 : Contact incliné

Deux problèmes surgissent quand on désire poser un objet sur la face inclinée de cette primitive :

- déterminer comment poser sur cette surface. Nous proposons une méthode qui consiste à décomposer le mouvement en trois étapes (voir la figrue 9) :

1ère étape : amener l'objet à poser A au dessus de la primitive B, par une translation T_0 (voir la figure 6) ;

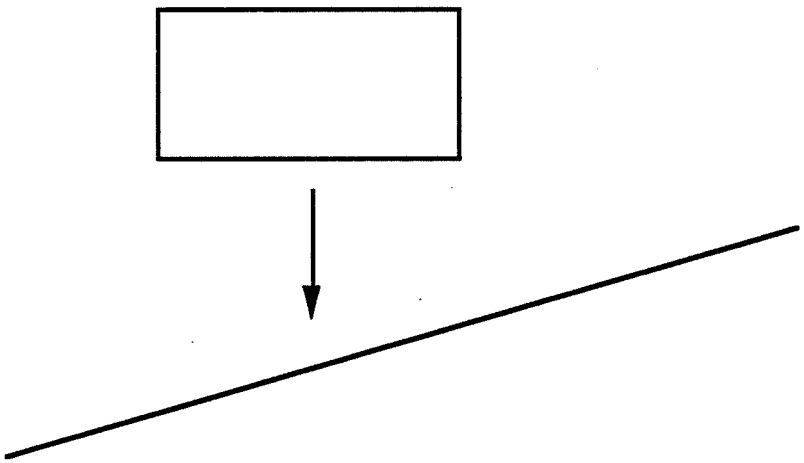


Figure 6 : Situation après la translation T_0

2ème étape : par une translation T_1 , amener A au contact de B (voir la figure 7) ;

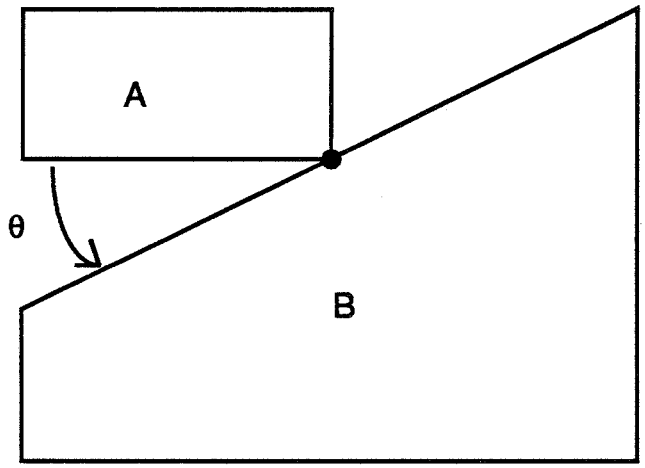


Figure 7 : Situation après la translation T_1

3ème étape : effectuer une rotation R_1 d'angle θ (voir la figure 8) ;

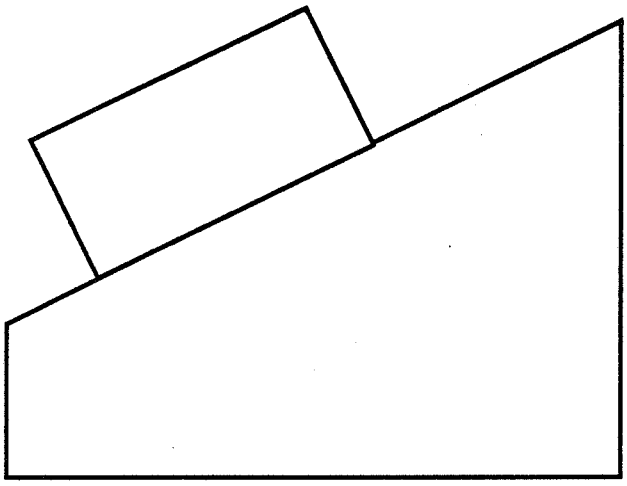


Figure 8: Situation après la rotation R_1

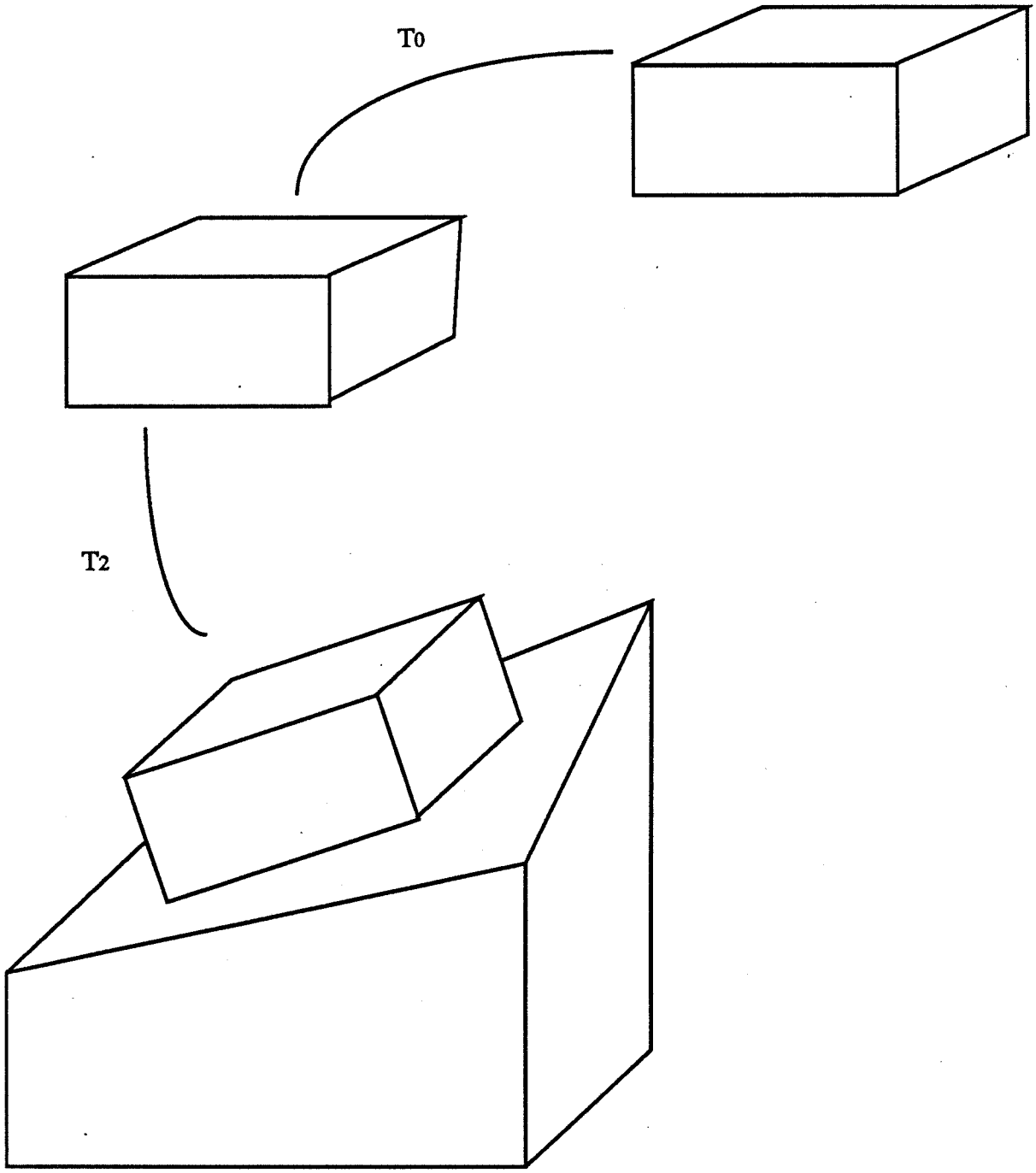


Figure 9 : $T_2 = f(T_1, R_1)$

- déterminer l'équilibre de l'objet posé. La loi de la physique portant sur l'équilibre d'un objet A posé sur un objet B est la suivante :

$$m\vec{g}_A + \vec{R}_B = \vec{0}$$

La détermination de la réaction $\vec{R}_B$ de l'objet B est purement expérimentale et dépend du type de matériau (bois, fer, plastique, ...). Les données des coefficients de frottement sont donc nécessaires, relatifs à chaque type de matériau.

Nous avions donc deux démarches :

- introduire des données expérimentales et estimer l'équilibre de l'objet A ;

- considérer que, dans tous les cas, l'objet A glisse.

La première démarche nécessite un nombre important de données et surtout savoir les interpréter. Jusqu'à quel angle d'inclinaison un morceau de bois tient-il sur une plaque de fer ? Encore faut-il savoir de quel type de bois. Aucun choix heuristique (nous en avons déjà pris pour d'autres problèmes) ne nous semblait justifié. Une étude poussée de ce phénomène s'impose.

La deuxième démarche est simple mais présente peu d'intérêt.

Aussi, dans la première version de notre interface, nous ne considérerons aucune surface inclinée.

• **les contacts non plans**, définis à partir des opérations booléennes différence et intersection.

Les opérations booléennes différence et intersection peuvent créer divers contacts possibles.

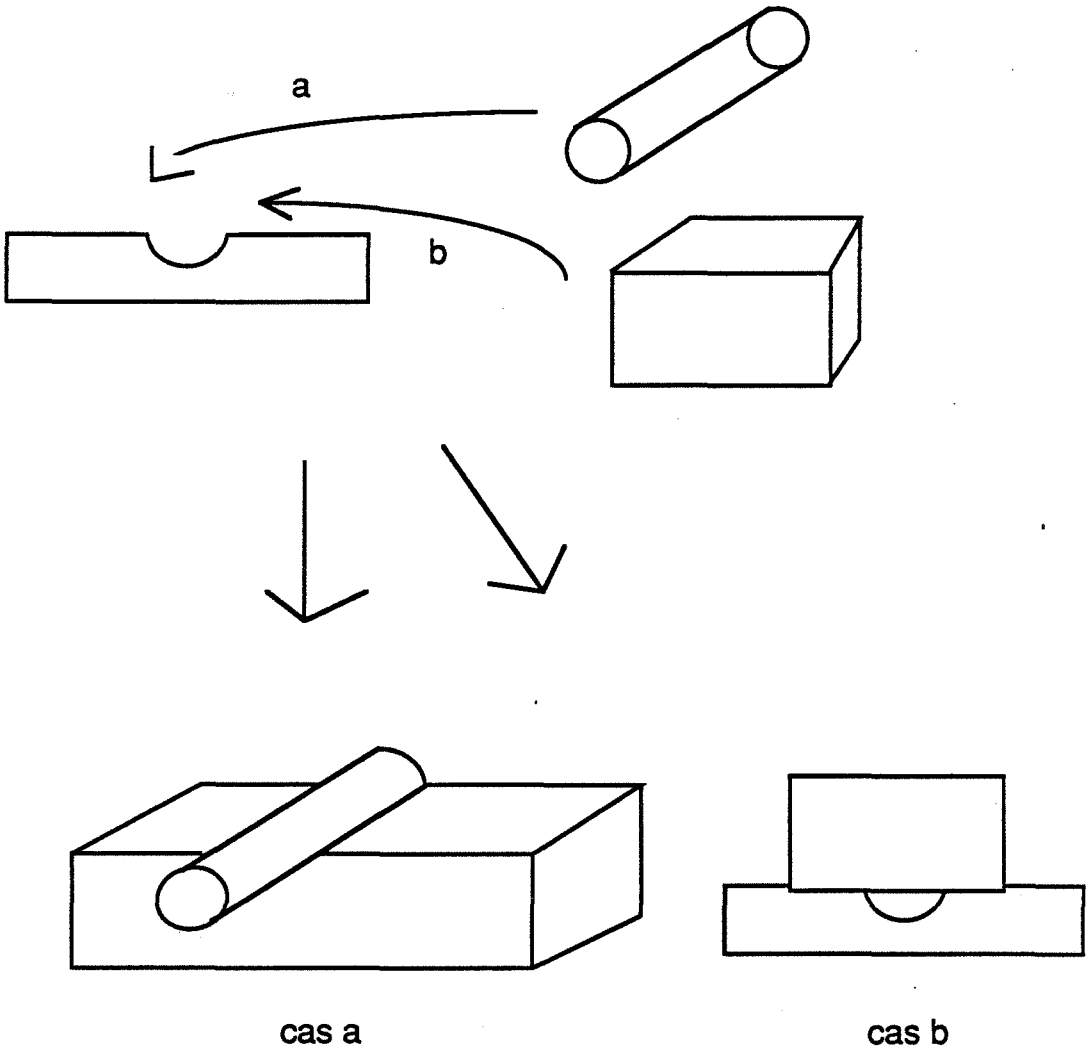


Figure 10 : - cas a - l'objet à poser s'emboîte parfaitement - cas b - la cavité d'emboîtement ne peut être utilisée

La pose d'un objet sur de tels contacts (voir la figure 10) nécessite une étude assez poussée. Deux méthodes existent :

- une connaissance exacte de la géométrie des contacts permet de résoudre ce genre de problème. De nombreux travaux en robotique existent. Cependant, Laugier ([LAUG 87]) précise qu'une restriction des contacts est également nécessaire. Il se limite, dans un premier temps, aux contacts qui mettent en jeu uniquement des surfaces planes. La figure 11 en présente des exemples.

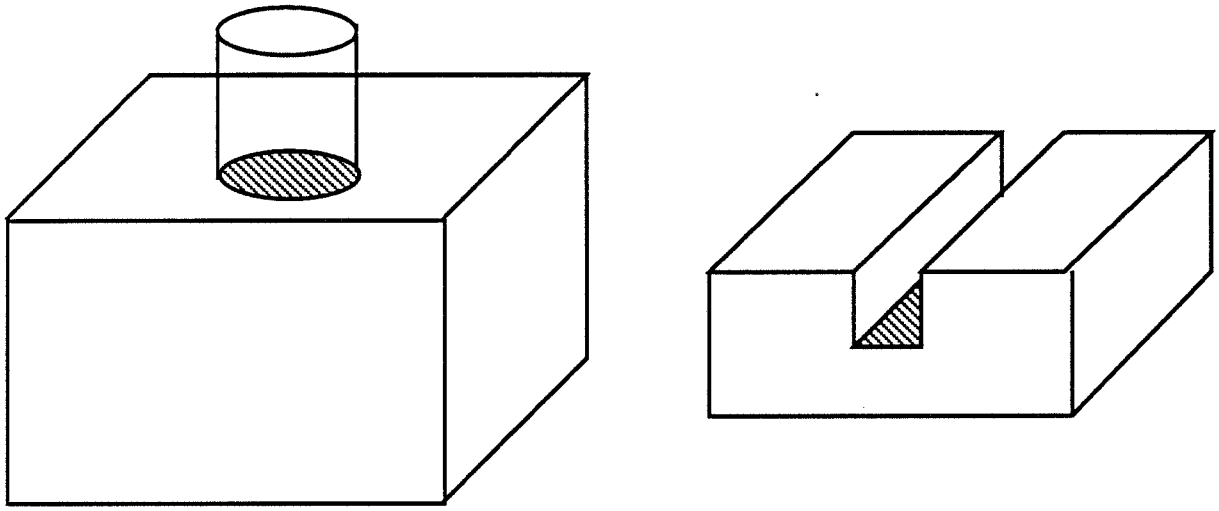


Figure 11: Quelques exemples de contacts

- la modélisation déclarative permet de concevoir et d'assembler divers éléments à partir de leur propriétés. Les techniques de l'intelligence artificielle rendent possible cette approche, les travaux de Martin ([MART 88]) sur la modélisation des polyèdres, en sont un exemple.

Notre application, dans sa première version, ne peut prétendre

- avoir une connaissance exacte des contacts. Une deuxième représentation par facettes serait alors nécessaire ;

- introduire au niveau de chaque objet des propriétés géométriques que déterminerait, au fur et à mesure, un moteur d'inférence. L'interactivité en souffrirait certainement.

Ainsi, dans une optique de faisabilité, en nous limitant aux seules primitives cube, sphère et cylindre, notre restriction porte sur deux aspects :

- seule l'opération booléenne union est autorisée ;

- les divers types de contacts possibles sont :

  - un point

  - un segment

  - une face

4.2.4 Un exemple

L'exemple de la figure 12 permet de résumer les notions introduites ci-dessus.

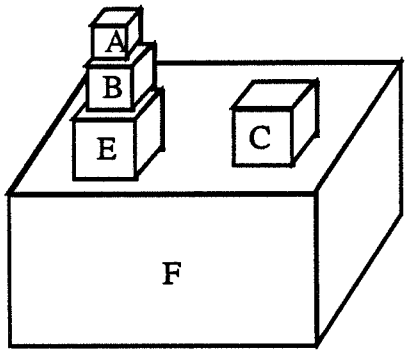


Figure 12

Cette scène est représentée par les objets suivants :

O1 :
fils : [A, B]
opbool : union
contact_haut : A
contact_bas : B
contrainte : [poserSur1]
methode :
deplacer_vers
modifier_taille
poser_sur

O2 :
fils : [E, TMP1]
opbool : union
contact_haut : (E, C)
contact_bas : F
contrainte : [poserSur1]
methode:
deplacer_vers
modifier_taille
poser_sur

A :
fils : [ ]
contact_haut : face1
contact_bas : face2
contrainte : [êtreSur2]
père : O1
méthode :
déplacer_vers
modifier_taille
poser_sur

TMP1:
fils : [C, F]
opbool : union
contact_haut : ()
contact_bas : ()
contrainte : [ ]
père : O2
methode :
deplacer_vers
modifier_taille
poser_sur

poserSur1 :
primitive1 : A
primitive2 : B

4.3 Les démons

Tous les attributs ci-dessus sont initialisés par défaut lors la création de l'instance. Certains attributs sont cependant liés à d'autres attributs d'instances. Le *poids*, le *centre de gravité*, la *boîte englobante* d'une instance *objet scène* constituée de deux *primitives* est la somme ou la combinaison des attributs respectifs des deux *primitives*. La structure arborescente de l'objet (*fils*, *père*, *opération boléenne*) ainsi que les attributs *contact_haut*, *contact_bas* doivent aussi être gérés automatiquement.

Des réflexes sont donc nécessaires pour propager toute modification d'un attribut sur ceux qui lui sont liés. Kool possède des descripteurs particuliers d'attribut, les *démons* (WhenFilled, WhenRemoved,...), qui contiennent la liste des fonctions à évaluer à la suite d'une manipulation sur l'attribut considéré.

Prenons l'exemple du poids. Le poids d'une instance *objet scène* est la somme des poids de toutes les *primitives* qui le composent. L'attribut *Poids* possède donc un descripteur *WhenFilled* (voir la figure 4) qui déclenchera la fonction Lisp calPoidSceObj :

```
(de calPoidSceObj (self attribute newValue)
 ;calcul du poids de l'objet scène
 ;on a modifié le poids de l'objet
 ;on doit répercuter cette modification
 ;sur son père, conjointement avec son frère
   (cond
      ((eq nil (getValue self Pere)) ())
      ((eq 2 (length (getValue (getValue self Pere) Fils)))
         (putValue
              (getValue self Pere)
              Poids
              (+ (getValue
                      (car (cdr (getValue (getValue self Pere) Fils)))
                      Poids)
                 (getValue (car (getValue (getValue self Pere) Fils)) Poids))))
      ((eq 1 (length (getValue (getValue self Pere) Fils)))
         (putValue (getValue self Pere) Poids newValue))))
```

Cette fonction utilise deux fonctions prédéfinies dans Kool qui sont :

- *putValue* qui fixe la valeur d'un attribut ou d'un descripteur d'objet ;

- *getValue* qui détermine la valeur d'un attribut ou d'un objet.

4.4 Les fonctions *modifier_taille*, *déplacer_vers* et *poser_sur*

Nous avons choisi de mettre en place ces trois premières fonctions car elles paraissent d'un maniement naturel aux utilisateurs, ne demandant à ces derniers que de désigner (avec la souris) ou de nommer les objets. Elles permettent déjà la création de scènes complexes et, de plus, elles se complètent mutuellement.

Par ailleurs, elles semblent poser suffisamment de problèmes pour permettre une étude de la faisabilité de la solution que nous proposons, ce qui est le but principal recherché de nos travaux actuels.

4.4.1 Généralités

La fonction *modifier_taille* permet de modifier les dimensions d'un *objet graphique* de la scène à l'aide d'affinités. L'utilisateur peut ainsi grossir ou diminuer un *objet graphique* selon que le coefficient est supérieur ou inférieur à un.

La fonction *poser_sur* s'applique à deux *objets scènes* ou à deux *primitives* (pas à un *objet scène* et une *primitive*). Le principe utilisé consiste à mettre en contact le contact_haut de l'objet sur lequel on pose et le contact_bas de l'objet que l'on pose.

La fonction *déplacer_vers* fait également intervenir deux objets graphiques que l'utilisateur aura désignés (en les cliquant ou en les nommant). Elle est constituée en fait de plusieurs fonctions :

- *rapprocher*, *éloigner* : ces deux fonctions permettent de rapprocher ou d'éloigner deux objets graphiques A et B en effectuant une translation de vecteur directeur (centre de gravité de A, centre de gravité de B) ;

- *déplacer vers* : avec une vue de dessus, l'utilisateur a la possibilité de déplacer un objet

  - vers la gauche ;
  - vers la droite ;
  - vers le haut ;
  - vers le bas ;
  - vers un point désigné sur l'écran ;

tout en maintenant la contrainte de contact.

Comme nous l'avons expliqué dans le second paragraphe, l'application d'une de ces trois fonctions doit nécessiter une réaction de la part du système. Cette réaction se traduit soit simplement par l'affichage de renseignements (voir la figure 13), soit par une suite d'actions provoquées automatiquement par le système. L'exemple de la figure 2 présenté dans le paragraphe sur les spécifications est réalisable. Nos objectifs, de ce point de vue, sont atteints.

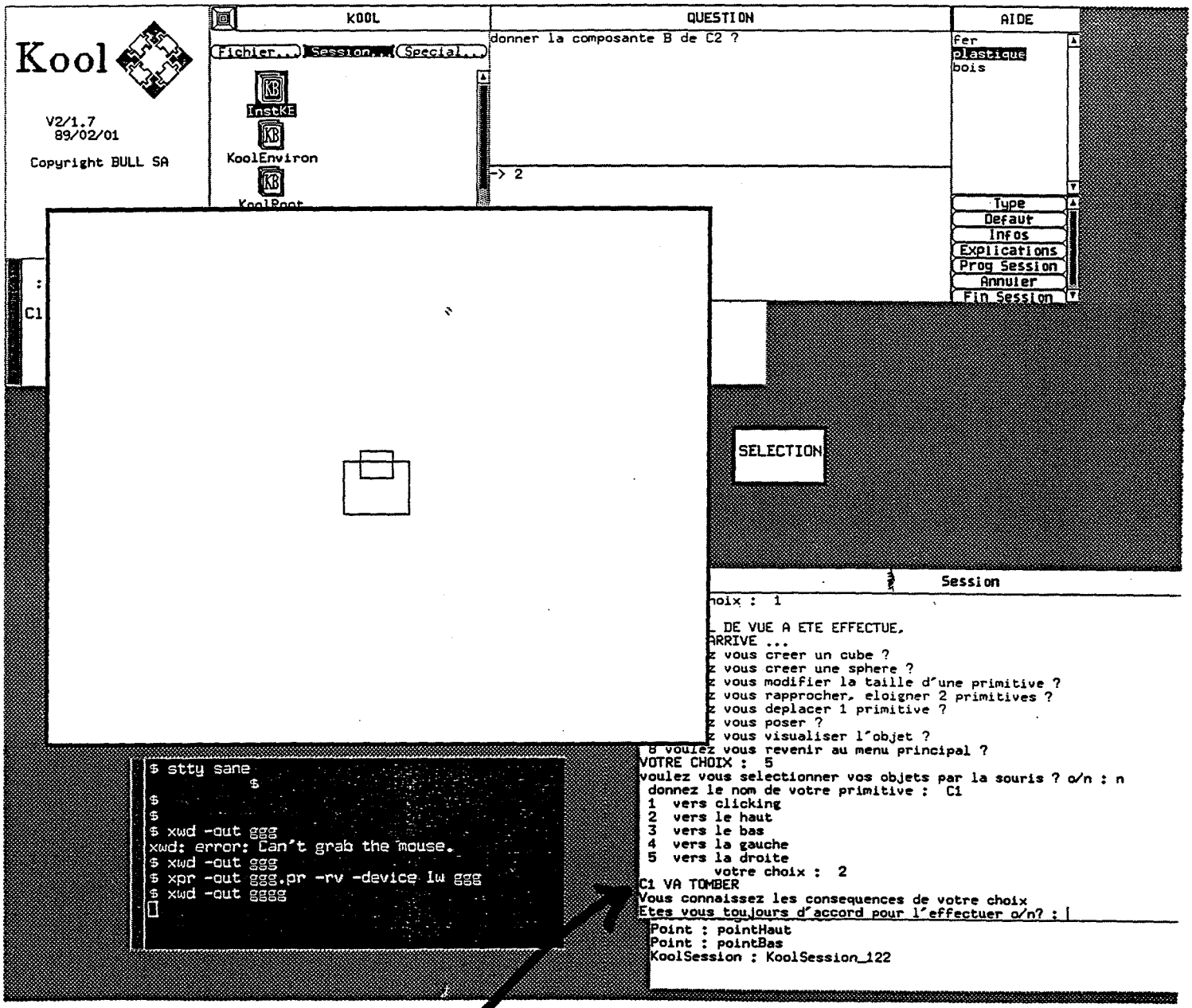


Figure 13 : grâce à une vue du dessus, l'utilisateur veut déplacer le petit cube (posé sur un gros) vers le haut. Le système réagit et lui signale qu'il va tomber.

Il est important de souligner que toute réaction nécessite:

- de manipuler des copies des objets. L'utilisateur peut alors, en fonction des renseignements obtenus, annuler son opération ;

- de répercuter un éventuel mouvement sur les objets liés, ce qui se fait par envoi de messages. En cas d'empilement d'objets, si on bouge l'objet de base, celui-ci enverra uniquement un message à celui (ou ceux) directement posé sur lui. Ce dernier enverra à son tour un message aux objets directement posés sur lui,... Tout objet regardera ses contraintes et son intersection avec les autres objets de la scène, et ne répondra qu'après à l'objet qui lui a envoyé le message. C'est l'objet placé le plus haut qui répondra donc à celui placé un cran en dessous, ... Finalement l'objet initial que l'utilisateur désire déplacer, recevra un message uniquement de celui (ou ceux) posé directement sur lui. La réponse tient cependant compte des réponses des autres objets dans la mesure où une seule réponse négative suffit pour annuler l'opération, sauf si l'utilisateur décide de passer outre.

4.4.2 Les fonctions accessibles par des méthodes : *modifier_taille* et *déplacer_vers*

Les fonctions *déplacer_vers* et *modifier_taille* sont accessibles par des méthodes sur les classes. Selon qu'elles s'adressent à des *primitives* ou à des *objets scènes*, elles ne demandent pas toujours les mêmes vérifications ou calculs et ne génèrent pas non plus les mêmes réactions. Dans les deux cas cependant, l'attribut *contrainte* est utilisé afin de pouvoir gérer et contrôler le tout par le biais de messages (comme nous l'avons expliqué dans le paragraphe 4.4.1).

Nous allons maintenant fournir quelques précisions sur les fonctions *modifier_taille* et *déplacer_vers*. Ces dernières utilisent plusieurs procédures identiques ; ce sont :

init (A) : on modifie la taille de A par rapport à quoi ? L'existence d'une contrainte êtreSur permet de savoir par rapport à quel point on doit effectuer l'homothétie.

vérif1(A) : A sera-t-il trop lourd ? Nous envisagions, nous l'avons dit, d'introduire une certaine logique physique. Jusqu'à quelle limite peut-on poser un objet d'un certain poids sur un autre. Encore faut-il tenir compte du type de matériau. Nous proposons la règle, purement empirique : le poids de A ne doit pas être dix fois supérieur au poids de l'objet en relation avec A par une contrainte *poserSur*. Sinon, un message est envoyé à l'utilisateur pour le lui signaler.

vérif2(A) : A tombera-t-il ? Le nouveau centre de gravité de A est testé avec tout objet en relation avec A dans une contrainte *poserSur*. De même, un message avertit l'utilisateur si A va tomber (voir la figure 13).

vérif3(A) : A arrivera-t-il au contact d'un autre objet ? On teste toutes les intersections entre A (après l'avoir modifié) et les boîtes englobantes des autres objets de la scène. Dans le cas d'intersection, elles sont signalées à l'utilisateur.

act1(A) : modifie la boîte englobante, le centre de gravité, le poids de A

act2(A) : déplace récursivement ce qui est sur A (avec toujours le test pour savoir si on arrive au contact d'un autre objet). Pour cela, un message "déplacer_vers" est envoyé à tout objet en relation avec A par une contrainte *poserSur* (A sera le second attribut, qui correspond à l'objet récepteur). Les objets situés sur A recevront donc un message qui activera la méthode *déplacer_vers*. Cette même méthode lancera également un message aux objets situés dessus, avant de modifier l'objet considéré ; la récursivité s'enchaîne.

déplacer(A,TL) : teste si A, subissant une translation TL, ne touche pas, par l'intermédiaire des boîtes englobantes, d'autres objets. Dans le cas où il existe une contrainte du type contact, le déplacement se fait le long du contact, sinon on déplace A avec la tranformation calculée (voir la figure 14). On modifie ensuite son centre de gravité et sa boîte englobante.

dupliquer(A,B) : effectue une copie de A appelé B ; les relations entre les attributs sont maintenues, avec les nouveaux noms des instances.

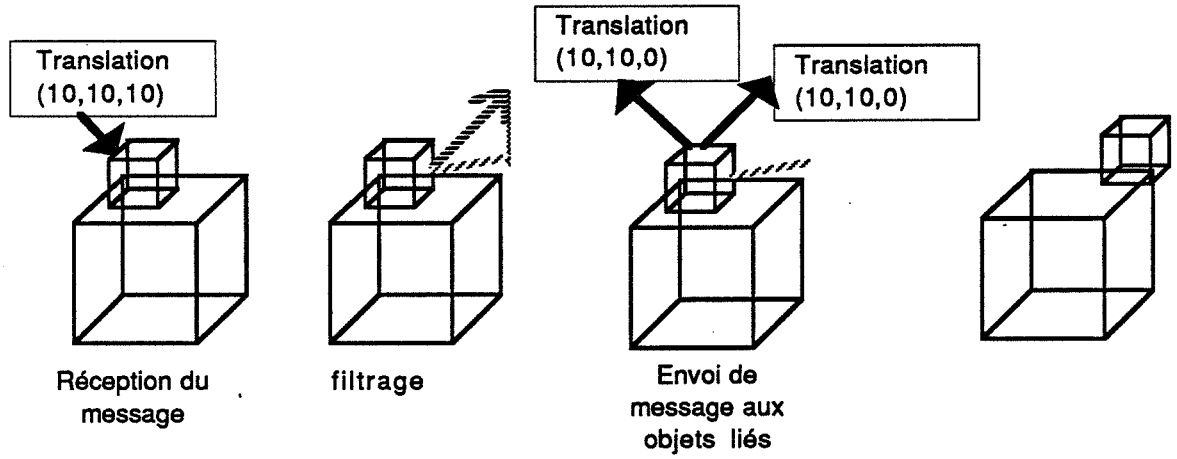


Figure 14 : Un exemple de déplacement contraint

• considérons maintenant la fonction *modifier_taille* : suivant qu'elle s'applique à une primitive ou à un objet, elle subit de légères modifications :

```
modifier_taille(objet scène)
début
dupliquer objet scène en TMP
vérif1(TMP)
        si OK1
            pour chaque primitive "primi" de TMP faire
              modifier_taille (primi)
            finpour
             vérif2(TMP)
                    si OK2
                            recopier TMP dans objet
                    fin OK2
        fin OK1
détruire TMP
fin


modifier_taille(primitive)
début
init(primitive)
vérif1(primitive)
  si OK1
        dupliquer père(primitive) en TMP
        vérif3(duplicata(primitive))
        vérif2(duplicata(primitive))
          si OK2
            vérif3(duplicata(primitive))
            act1(duplicata(primitive))
            calcul de la translation à faire subir aux objets posés sur primitive
            act2(duplicata(primitive))
                    si OK3
                      recopier TMP dans père(primitive)
                      recopier les objets posés sur primitive
                    fin OK3
            fin OK2
        détruire TMP
   fin OK1
fin
```

• Etudions maintenant la fonction *déplacer*-vers : les deux sous-fonctions "déplacer A vers B (éloigner, rapprocher)" ou "déplacer (vers cliquage, vers la droite, ...)" présentent le même algorithme, la seule différence étant au niveau du calcul du déplacement (dans la fonction *calcul*).

```
déplacer_vers(A,B)
        debut
        dupliquer A en TMP
        calcul de la translation TL
        vérif2(TMP)
        si OK1
                si (TMP est une primitive)
                        déplacer(TMP,TL)
                sinon   pour chaque primitive "primi" de TMP faire
                                déplacer (primi)
                        finpour
                finsi
                si OK2
                        act2(TMP)
                        si OK3
                                recopie des objets posés sur TMP
                                recopie de TMP dans A
                        fin OK3
                fin OK2
        détruire les objets posés sur TMP
        fin OK1
        détruire TMP
        fin
```

4.4.3 La fonction de contrainte : *poser_sur*

La seule fonction de contrainte que nous ayons implémentée, nous l'avons dit, est la fonction *poserSur*. L'application de cette fonction consiste à calculer la transformation nécessaire pour satisfaire la condition, à savoir "le premier objet est sur le second", puis d'instaurer une contrainte entre ces deux objets. Toute modification ou déplacement d'un des deux objets nécessitera alors de la part du système certaines vérifications ou transformations avant de les accepter.

Poser un objet graphique A sur un autre objet graphique B se fera de la façon suivante :

- une instance *poserSur* sera créée avec comme attributs A et B. Les attributs *contraintes* de A et B contiendront la relation *poserSur* ainsi créée.

- A sera déplacé. Le principe consiste à calculer la tranformation pour amener le *contact_bas* de A au *contact_haut* de B. Ce problème peut devenir vite complexe et a nécessité de notre part certains choix (voir le paragraphe 4.4.3.1) pour arriver à une première solution que nous détaillerons dans les paragraphes

suivants. Ensuite cette transformation est répercutée, toujours par des messages, à tous les objets liés à A. Comme pour les fonctions *déplacer_vers* et *modifier_taille*, le système signale tout risque d'intersection ou de perte d'équilibre. En cas d'accord de l'utilisateur, le déplacement aura alors lieu. Tout ceci est réalisé par la méthode *contact* . Décrivons la plus en détail, selon qu'elle s'adresse à deux *objets scènes* (i.e un objet composite) ou à deux *primitives*. La procédure *évolution_contact* modifie les contact_haut et contact_bas de A et de B, afin de conserver ou de créer les relations.

•***contact_objet***(A,B)

debut

dupliquer A en TMP
vérif1(TMP)
vérif2(TMP)
contact_pri(contact_bas(TMP),contact_haut(B))
déplacer toutes les primitives de TMP autre que le contact_bas
si OK1
    ;la translation TL doit être effectuée sur les objets posés
    ;sur A
    act2(TMP)
    si OK2
        recopier les objets posés sur TMP
        recopier TMP dans A
    fin OK2
détruire les objets posés sur TMP
fin OK1
détruire TMP

fin

• ***contact_pri***(A,B)

debut
vérif1(A)
vérif2(A)
si OK1
    calcul de la translation TL de A vers B
    déplacer(A,TL)
    évolution_contact(père(A))
fin OK1
fin

4.4.3.1 Les choix effectués

Nous avons précisé dans le premier paragraphe la définition des attributs *contact_x* (x = haut ou bas) restreints, pour les besoins de l'étude actuelle, à des points, des segments ou des surfaces planes.

Cependant l'attribut *contact_x* d'une instance d'*objet scène* peut être constitué de plusieurs primitives. Il peut aussi avoir plusieurs composantes connexes, lesquelles peuvent ou non être situées à la même hauteur (voir la figure 15), ...

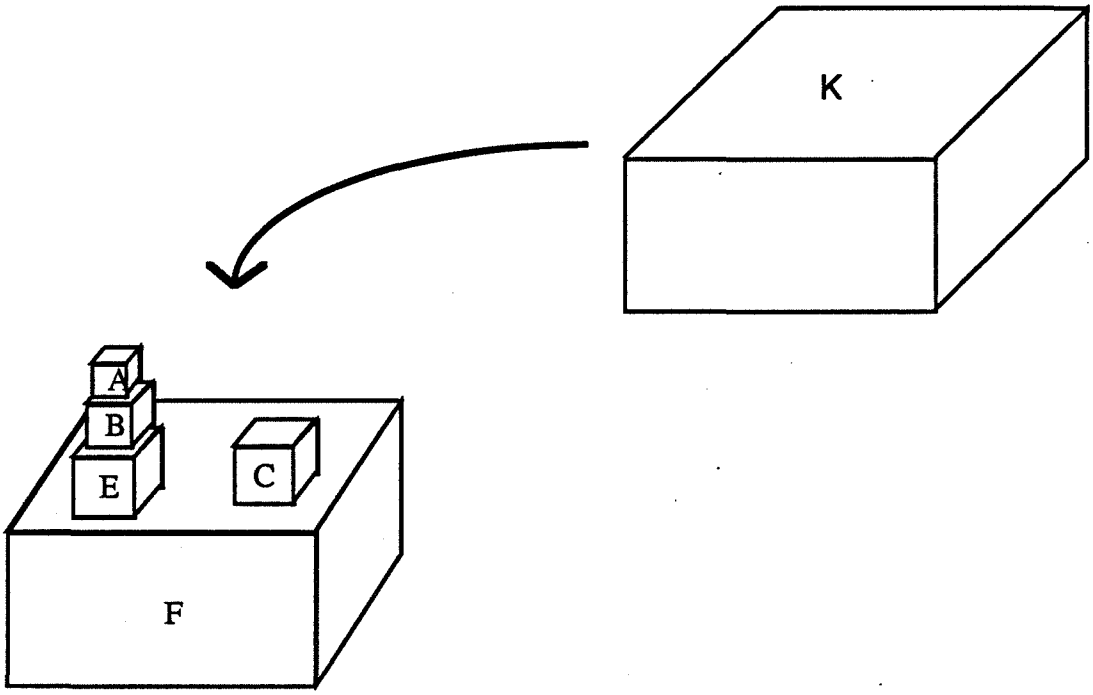


Figure 15 : l'objet K reposera-t-il sur A, C, A et C, ... ?

Il se dégage alors un certain nombre de cas que le système devra étudier. Analysons les. Des hypothèses seront émises au fur et à mesure de cette étude, afin de restreindre les possibilités :

1ère hypothèse :

{
Nous n'étudierons que le cas de la pose d'un objet à un seul *contactBas* sur un objet à p *contactHaut* ou sur p objets. Le résultat de cette étude permettra de mesurer la faisabilité de l'extension suivante : poser un objet à n *contactBas* sur p *contactHaut*, d'un ou plusieurs objets (par exemple poser une chaise sur quatres cubes).
}

L'analyse porte maintenant sur le *contactHaut* de l'objet receveur. Celui-ci (le *contactHaut*)

- peut ou non être connexe ;

- est constitué de composantes connexes qui sont situées ou non à la même hauteur.

Divers cas en découlent. Enumérons les :

• **1er cas** : le *contactHaut* est composé d'une seule primitive (donc connexe et à la même hauteur). Ce cas est simple et se résoud de façon triviale.

• **2ème cas** : l'objet possède plusieurs composantes connexes qui sont à la même hauteur.

2 ème hypothèse :

{
Nous avons établi des zones de contact (la face inférieure pour le cube, ...). Poser revient à mettre en contact ces deux zones, aussi les solutions de la figure 16 ne seront pas proposées par le système.
}

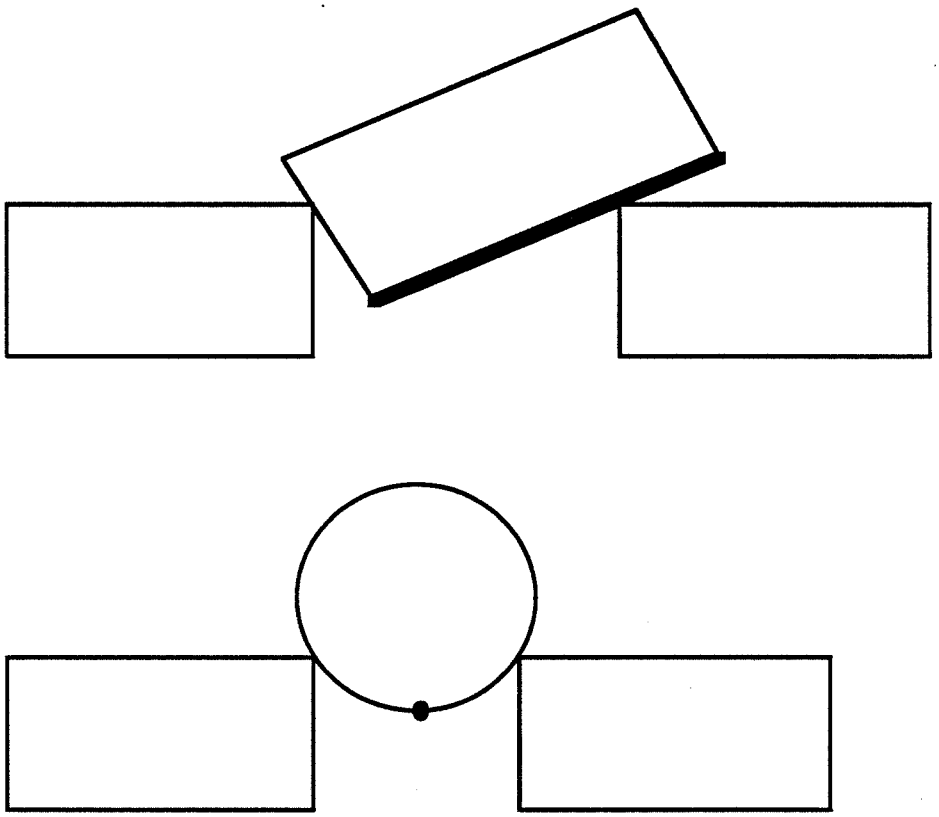


Figure 16 : Deux exemples de possibilités non proposées par le système

Prenons le cas de la figure 17 qui constitue une vue de dessus d'un objet A, composé de deux cubes, d'une sphère (le point) et d'un cylindre couché (le segment). Poser un objet B sur cet objet A pose en fait deux problèmes :

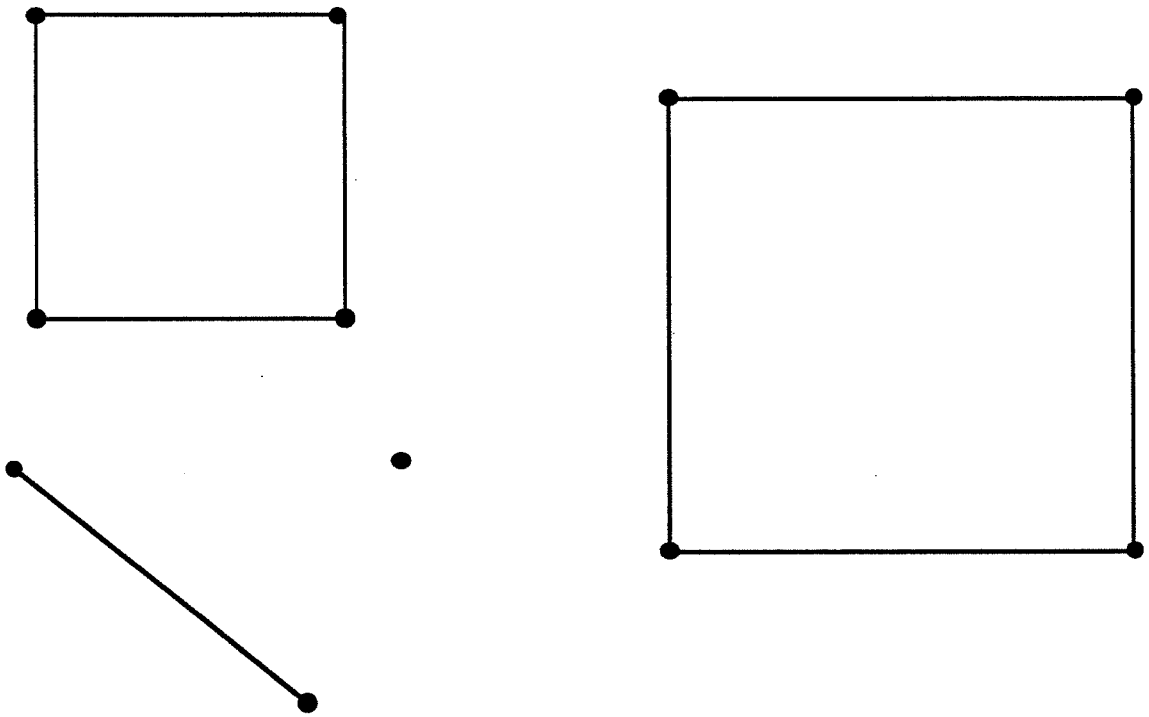


Figure 17 : Vue de dessus du *contactHaut* de plusieurs objets destinés à recevoir un autre objet

problème 1 : trouver un ensemble de points sur lesquels poser l'objet ;

problème 2 : poser A sur cet ensemble de points de façon à ce qu'il soit en position d'équilibre.

Comment arriver à déterminer un bon ensemble de points. Comment positionner l'objet B sur cet ensemble. Diverses orientations de l'objet B conviennent, une est certainement meilleure que les autres. L'objet peut-il reposer sur une composante, sur deux composantes, ..., sur chacune des composantes. Plusieurs hypothèses s'imposent : l'interactivité est, ne l'oublions pas, **impérative**.

3 ème hypothèse :

{
L'utilisateur a la possibilité de désigner toute primitive ou tout objet de la scène. Aussi, si l'utilisateur demande de poser un objet A sur un objet B dont le *contact_haut* est constitué de plusieurs primitives, c'est dans le but de faire reposer A sur **toutes** les primitives constituant le *contact_haut.*
}

4 ème hypothèse :

{
Le système doit être en mesure de pouvoir séparer assez rapidement (interactivité oblige) les situations qui autorisent une solution, des situations qu'il ne peut pas résoudre de manière autonome. Si c'est possible, le système proposera alors une solution standard dans un processus entièrement **automatique**. Si l'utilisateur souhaite modifier cette position, il peut le faire à l'aide de fonctions comme *déplacer*. L'aide du système consiste donc à fournir une solution par défaut, assurant par là même la stabilité de l'objet posé ainsi que la cohérence du modèle. Par contre, face à une situation trop complexe pour laquelle il ne peut proposer une solution, le système signale à l'utilisateur qu'il ne peut satisfaire sa demande. Nous nous dirigeons alors vers une solution **semi-automatique** au cours de laquelle s'établit un dialogue entre l'utilisateur et le système pour arriver à trouver une solution.
}

Notre démarche consiste donc à déterminer si une solution entièrement automatique est possible ou non. Nous verrons plus en détail ultérieurement comment est effectuée cette étude. Nous verrons de même la solution standard proposée aux problèmes 1 et 2.

• **3ème cas** : l'objet receveur possède un contactHaut (connexe ou non) qui n'a pas partout la même hauteur (la figure 15 en est un exemple). Poser un objet sur un contact de ce type en respectant l'hypothèse 3, à savoir faire en sorte de faire reposer l'objet sur toutes les composantes, est mathématiquement complexe. Aussi, dans un premier temps, nous estimons que le système ne peut proposer une solution dans les délais des compatibles avec l'interaction. L'approche semi-automatique s'impose.

Devant la combinatoire du problème, nous avons donc été amenés à établir certains choix et à faire certaines hypothèses. Nous allons exposer dans les paragraphes qui suivent les solutions proposées.

L'existence d'une représentation des connaissances par des règles dans Kool et les divers travaux étudiés dans le paragraphe 2 nous ont conduit à représenter les connaissances opératoires sous forme de règles que nous détaillerons dans le paragraphe 4.4.3.2. Nous exposerons alors, en conclusion, les avantages de ce type de représentation des connaissances, tels qu'ils nous sont apparus.

4.4.3.2 Les règles de satisfaction de contrainte

L'utilisateur formule la demande "poser A sur B". Une instance *poserSur* est créée avec comme attributs A et B. Cette création va alors déclencher des règles qui vont, en

fonction de la situation, proposer une solution automatique ou semi-automatique. Nous allons décrire dans le paragraphe 4.4.3.2.1 la hiérarchie *poserSur* et ses sous-classes ainsi que leur utilisation par le système. Nous détaillerons ensuite les approches automatique et semi-automatique dans le paragraphes 4.4.3.2.2.

4.4.3.2.1 La classe poserSur

La classe *poserSur* (voir la figure 18) possède, en plus des deux attributs *objetàPoser* et *objetReceveur* (voir le paragraphe 4.2.2), un attribut *possible*. Ce dernier, un booléen, est le résultat de l'analyse du système face à la situation. Il vaut **vrai** quand le système estime que le problème est soluble de façon automatique, **faux** sinon. Cet attribut possède alors un descripteur WhenFilled (un démon) qui

- déclenchera le calcul du positionnement et la répercussion aux objets liés, si possible = vrai

- enverra un message à l'utilisateur et génèrera la démarche semi-automatique, si possible = faux

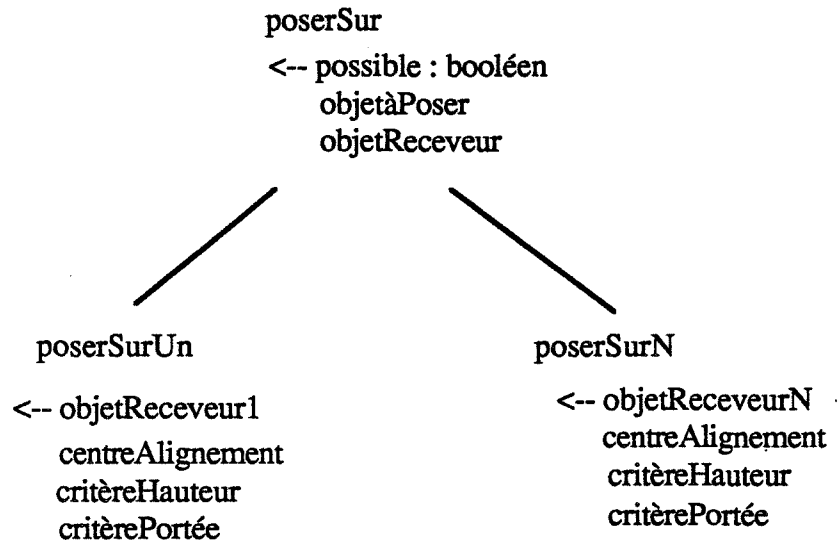


Figure 18 : La sous-hiérarchie PoserSur

Par démon donc, nous déclencherons les calculs. Par les règles, nous allons établir la valeur de *possible*. Cela nécessite alors de définir plusieurs sous-classes de *poserSur* : *poserSurUn*, *poserSurN* (voir la figure 19), ...

```
{InstClass : poserSurN                    critereHauteur : ()
Super : Relation                                Mode : Mono
<--     objetReceveurN : ()                     Type : [true,false]
                Mode : Multi              criterePortee : ()
                Type : GraphicalObject          Mode : Mono
        CentreAlignement : ()                   Type : [true,false]
                Mode : Mono               ->}
                Type : Point
                ToCompute : calCentreAlignement
```

Figure 19 : La classe PoserSurN

Poser un objet sur 1,2,3, ... composantes connexes s'effectue selon le même principe : on définit un point (appelé *centre d'alignement*) sur lequel sera aligné le centre de gravité de l'objet à poser. Selon le nombre de composantes, le calcul et l'existence (le but est de savoir si on peut poser) de ce point changent, d'où la nécessité de ces sous-classes. Décrivons-les de façon plus précise avant de présenter les différentes méthodes : automatique dans le cas où c'est possible, et semi-automatique en cas d'échec.

• *poserSurUn* possède, en plus des attributs hérités de sa super classe *poserSur* :

- *objetReceveur1* (*OR1*) : qui correspond à l'unique composante connexe du *contact_haut* de l'objet receveur ;

- *centreAlignement* : dans le cas d'une seule composante, il s'agit de son centre de gravité ;

- *critèreHauteur* : booléen ;

- *critèrePortée* : booléen.

• *poserSurN* possède également, en plus des attributs hérités, plusieurs attributs :

- *ORN*: est la liste des composantes du *contact_haut* de l'objet receveur considéré ;

- *centreAlignement* : il s'agit du barycentre des centres de gravité des composantes ;

- *critèreHauteur* : booléen,

- *critèrePortée* : booléen.

Le problème consiste maintenant à déterminer la valeur de l'attribut *possible*. Dans le cas où *possible* est vrai, le démon associé déclenchera alors la fonction de positionnement associée.

4.4.3.2.2 Les approches automatique et semi-automatique

A la suite de notre étude dans le paragraphe 4.4.3.1, il est possible, de façon générale, de poser un *contactHaut* d'un objet A constitué d'une seule composante connexe sur n composantes connexes si les deux conditions suivantes sont respectées :

> **condition1** : les composantes connexes sont à la même hauteur : *critèreHauteur* est alors vrai.
>
> **condition 2** : l'objet A, une fois aligné avec le centre d'alignement de l'ensemble des composantes, repose sur chacune des composantes : *critèrePortée* est alors vrai.

Cela se traduit en Kool par la règle en chaînage arrière suivante :

```
{Rule - : Générale
If  *poserSur = *P
    *P^critèreHauteur
    *P^critèrePortée
then
        *poserSur^possible = true
}
```

Considérons maintenant l'analyse des conditions 1 et 2. Elles se feront par l'intermédiaire des règles.

> • **condition 1 :** cette condition est simple à tester. Dans le cas d'une instance *poserSurUn*, *critèreHauteur* est automatiquement vrai. Dans le cas d'une instance *poserSurN,* il suffit de comparer les hauteurs des éléments de la liste *ORN*. Donnons l'exemple de la règle qui traite le cas de *poserSurUn* :

```
{Rule - : poserSurUn
If  *poserSur = *P
    *P isa poserSurUn
   then
       *poserSur^critèreHauteur = true
}
```

• **condition 2** : l'analyse de cette condition est plus délicate. La recherche de la valeur de l'attribut *critèrePortée* déclenchera un démon *recouvrement*. *Recouvrement* est une méthode qui va tester s'il est possible de poser notre objet sur les composantes connexes considérées. Elle affectera la valeur vrai au booléen dans le cas de l'affirmative. Il est important de souligner qu'elle diffère selon qu'elle s'adresse à des instances *poserSurUn*, *poserSurN*, ... Là encore (pour *centreAlignement* aussi), la hiérarchie objet est bénéfique : à chaque niveau de la hiérarchie, on associe une méthode, de même nom, mais générant des calculs différents. Détaillons cette méthode selon les instances auquelles elle se rapporte :

- *poserSurUn* : le calcul est simple : critèrePortée est toujours vrai dans la mesure où on a aligné le centre de gravité de l'objet à poser avec le centre de gravité de l'unique composante.

- *poserSurN* : reprenons nos hypothèses émises auparavant :

  • nous souhaitons poser une composante connexe sur n composantes connexes situées à la même hauteur ;

  • une sphère ne peut reposer sur plusieurs composantes connexes (hypothèse 2) ;

Emettons également d'autres hypothèses :

• nous ne possèdons pas de fonction permettant de "coucher" une primitive, aussi tout cylindre sera considéré comme "debout" ;

• l'objet à poser est soit un cube, soit un disque (dans le cas du cylindre debout) ;

• les composantes connexes réceptrices sont soit des cubes, soit des sphères ;

• tout cube a ses cotés parallèles aux axes, aucune fonction ne rotation n'étant pour l'instant permise.

Finalement le problème se ramène à l'intersection planaire suivante :

Données :

$O_1, O_2, ..., O_n$: n rectangles (représentatifs des *contactHauts* des cubes) ou points (représentatifs des *contactHauts* des sphères)

A : un objet rectangulaire ou circulaire (représentatif du *contactBas* de l'objet à poser)

Question :

l'objet A intersepte-il tous les objets O_1, O_2, ..., O_n suivant une certaine direction α pour le cas du rectangle A (voir la figure 20)

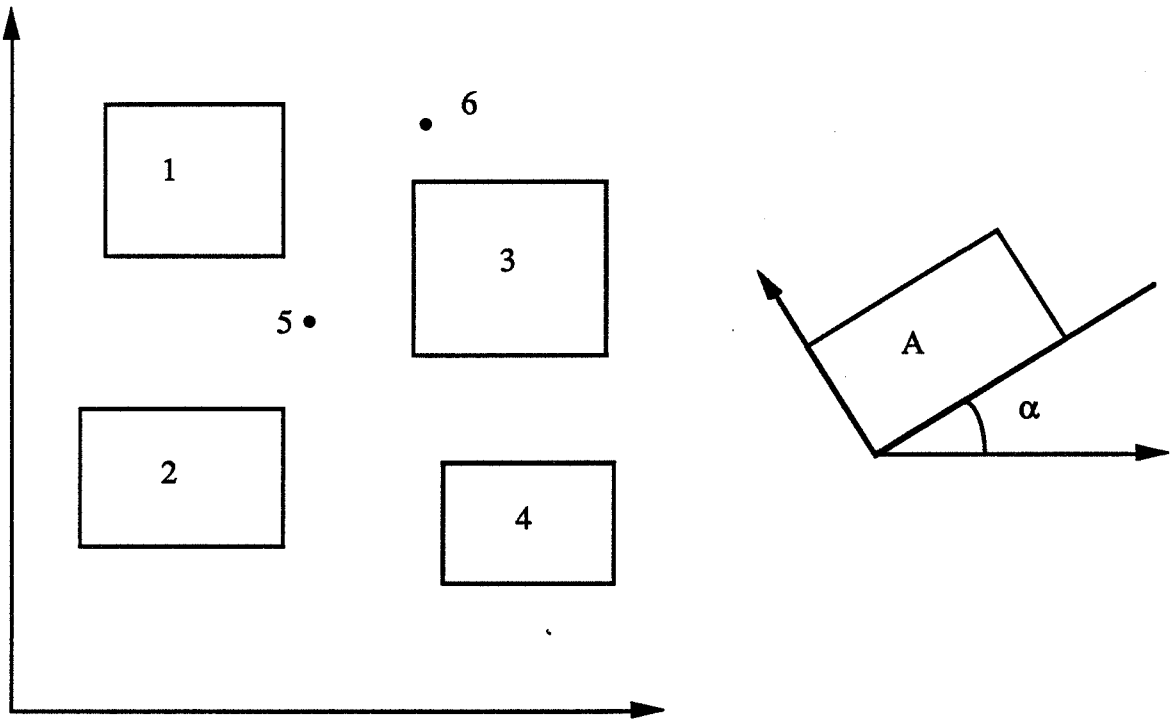


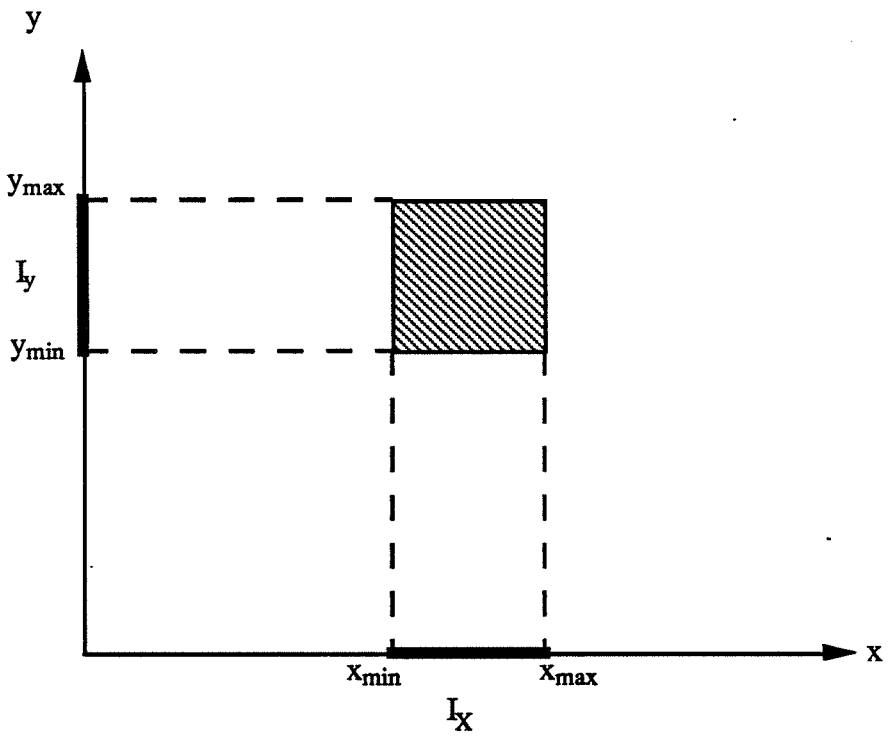
Figure 20 : L'objet A, à poser, est orienté suivant une direction α

Examinons le cas où $\alpha = 0$: (tous les objets rectangulaires ont les cotés parallèles aux axes)

Définition 1 : on appelle *rectangle critique*, le rectangle défini par le produit d'intervalle $I_x \times I_y$ avec

$I_x = [x_{min}, x_{max}]$ si $x_{min} < x_{max}$

$I_y = [y_{min}, y_{max}]$ si $y_{min} < y_{max}$



avec (voir la figure 21)

$x_{min} = \min x_d\,(i)_{1 \le i \le n}$ $\quad x_{max} = \max x_g\,(i)_{1 \le i \le n}$

$y_{min} = \min y_h\,(i)_{1 \le i \le n}$ $\quad y_{max} = \max y_b\,(i)_{1 \le i \le n}$

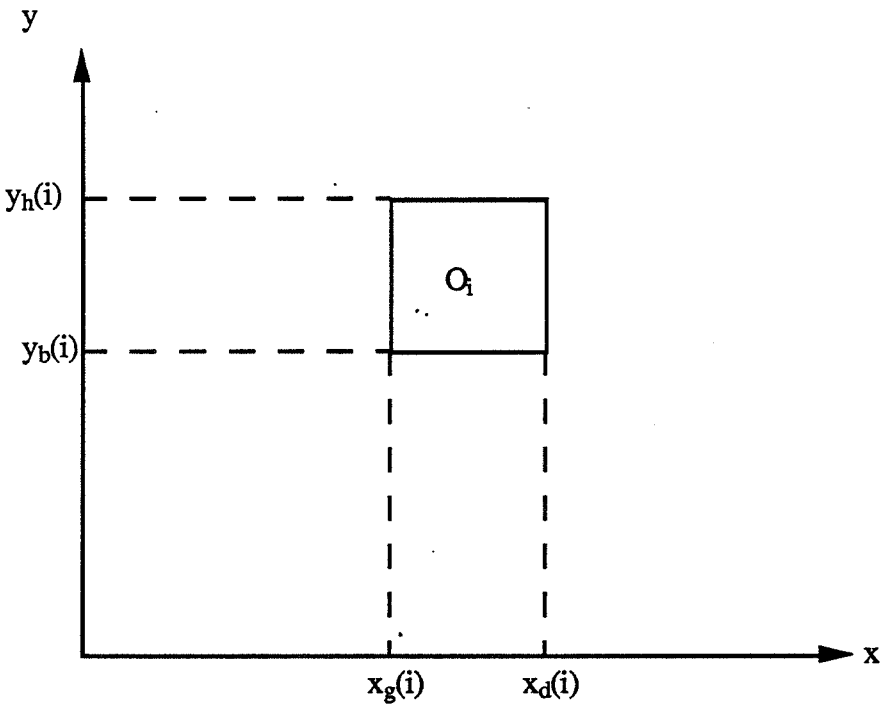


Figure 21

Prenons l'exemple de la figure 22 sur un ensemble d'objets

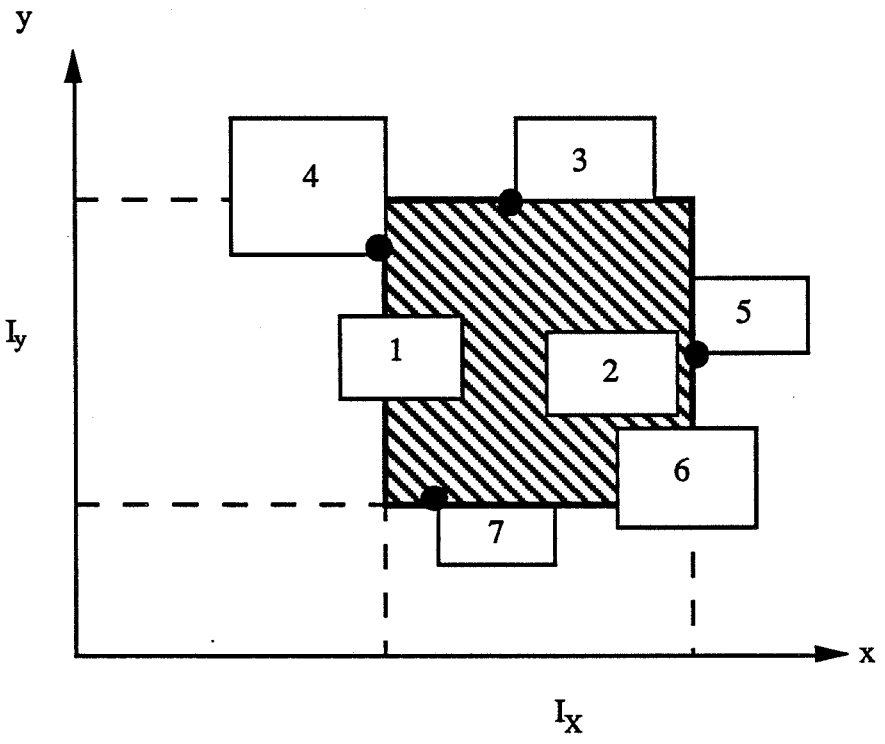


Figure 22

Remarques :

R1 : dans le cas où $x_{max} \leq x_{min}$ (voir la figure 23), tout objet d'épaisseur en x non nulle peut reposer sur toutes les composantes ; I_X se ramène à epsilon .

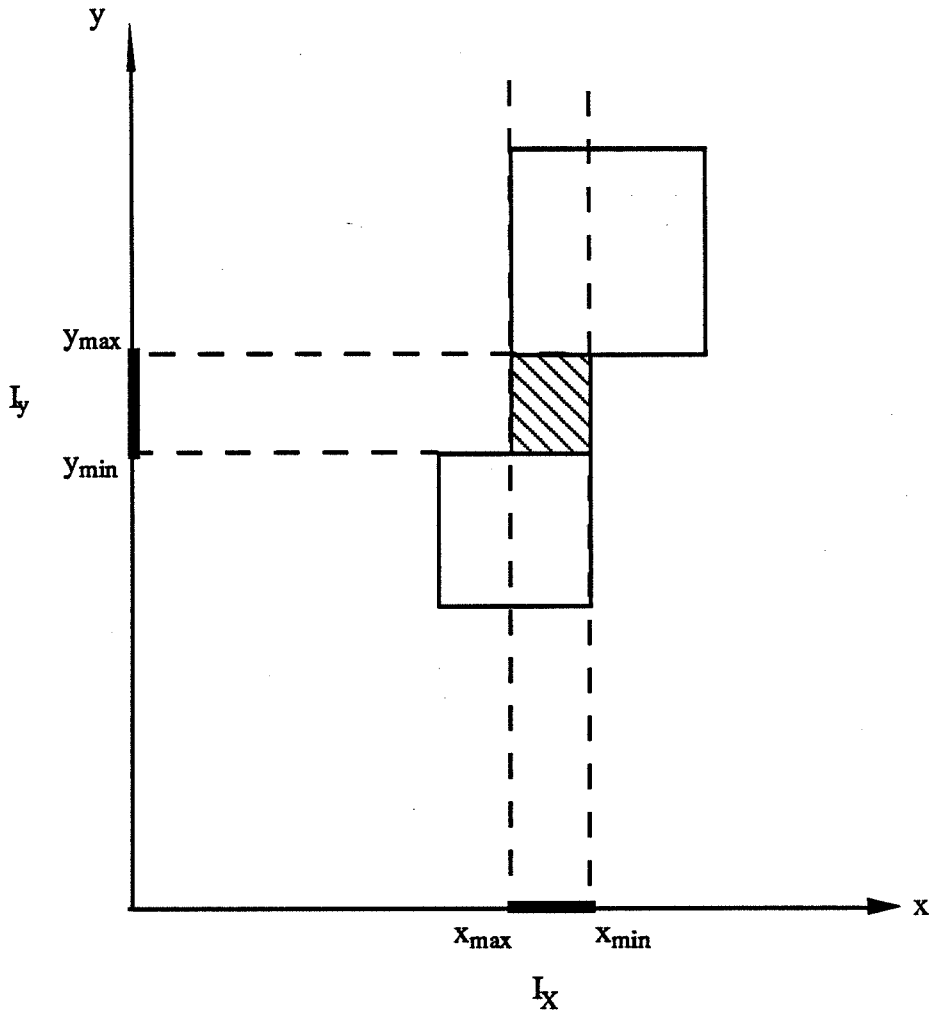


Figure 23

R2 : dans le cas où $y_{max} \leq y_{min}$ (voir la figure 24), I_y se ramène, de façon symétrique, à epsilon .

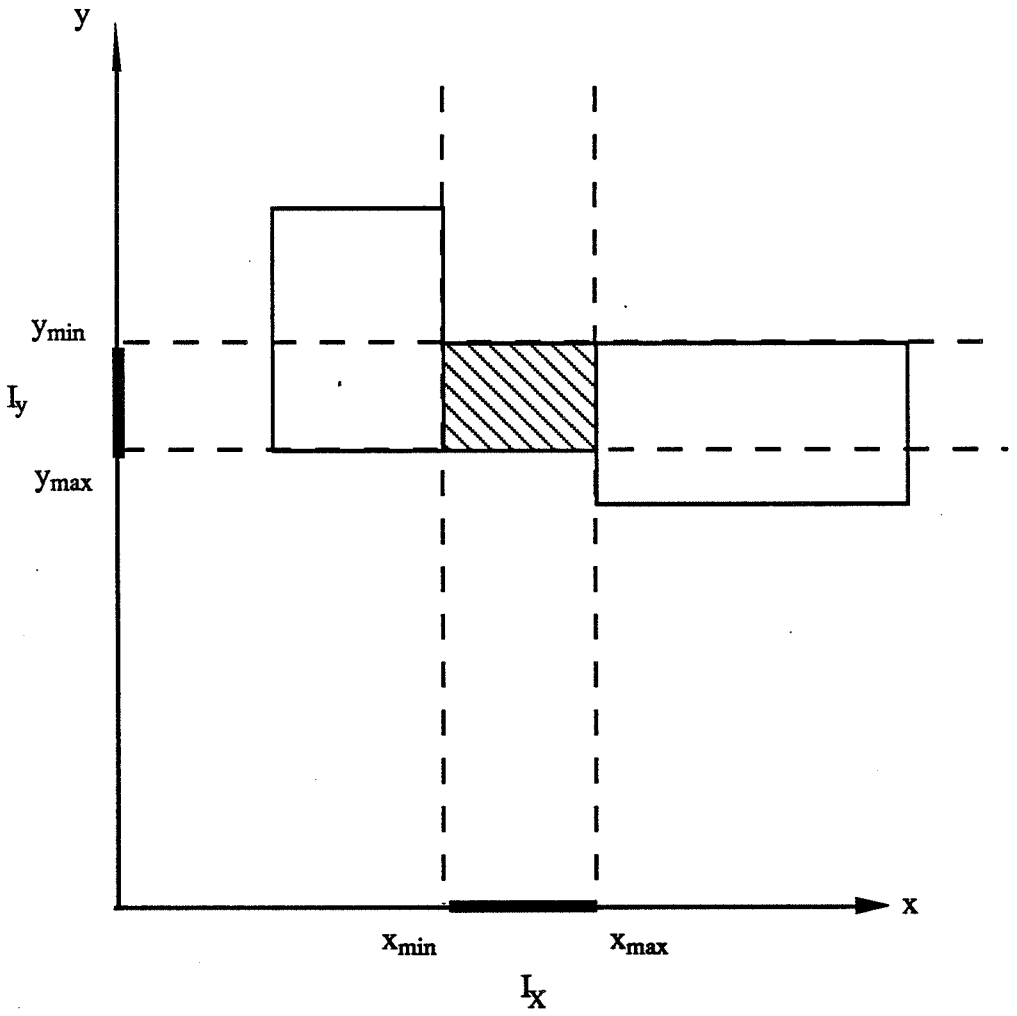


Figure 24

Définition 2 : nous dirons qu'un objet 1 recouvre strictement un objet 2 si l'objet 1 contient l'objet 2 en son intérieur (voir la figure 25)

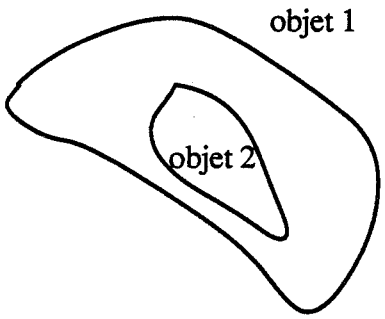


Figure 25

Théorème : une condition **nécessaire et suffisante** pour que l'objet A repose simultanément sur tous les objets O_1, O_2, ..., O_n est qu'il recouvre strictement le rectangle critique.

Corollaire : Il existe un algorithme en O(n) résolvant ce problème (on fait exactement 4(n-1) comparaisons)

• **Examinons le cas où $\alpha \neq 0$:**

Malheureusement, le théorème n'est plus vrai. Prenons l'exemple de la figure 26 ; le rectangle critique obtenu ne repose pas sur toutes les composantes.

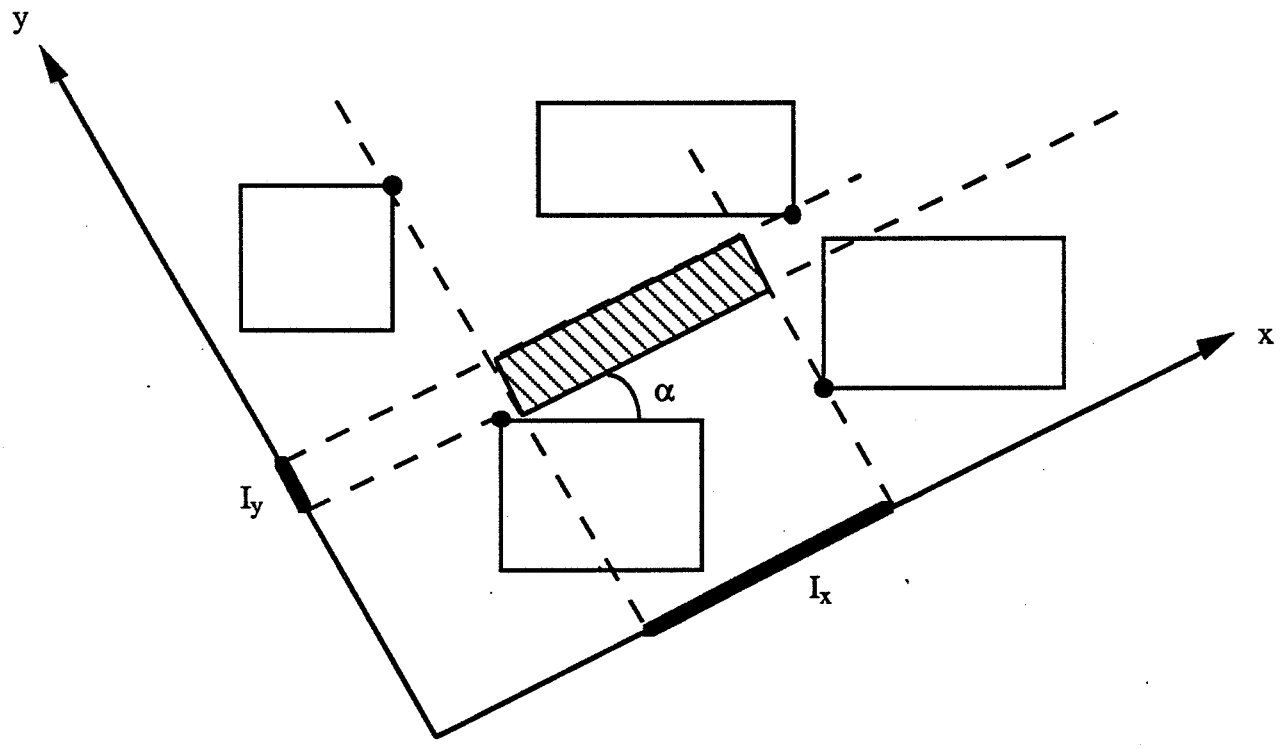


Figure 26 : $\alpha \neq 0$

Ce cas nécessite une étude complémentaire en vue de proposer une condition nécessaire et suffisante. Elle sera à envisager.

Dans le cas où les conditions 1 et 2 n'ont pas été remplies (ou plutôt ont contribué à donner une valeur fausse à l'attribut *possible*), l'approche automatique ne peut avoir lieu. Un dialogue s'instaure donc entre le système et l'utilisateur. Le système signale son impossibilité à satisfaire les volontés de l'utilisateur. Suite à l'échec, l'utilisateur a toutefois la possibilité de poser son objet sur quelques unes des composantes connexes (l'échec a eu lieu pour toutes les composantes, peut être en est-il autrement pour seulement 2, 3, ..., composantes) de l'objet receveur

- en nommant les composantes ;

- en positionnant approximativement et interactivement l'objet au dessus des composantes souhaitées : le système se charge de les détecter.

Le processus est relancé : le système analyse la situation. En cas d'échec, l'utilisateur aura encore la possibilité de proposer d'autres composantes, ou de restreindre leur nombre.

4.5 L'état actuel de la réalisation

Dans son état actuel, notre interface est composée de

- 15 classes décrivant la hiérarchie objet ;

- 15 règles nécessaires à la prise en compte automatique du système :

  - pour la détermination du poids en fonction du matériau,

  - pour poser un objet selon le nombre de composantes et la nature de l'objet receveur,

  - pour engendrer la démarche semi-automatique si besoin est ;

- 61 fonctions LeLisp (4695 lignes) ajoutées aux fonctions KOOL

  - pour créer les différents objets de la scène ;

  - pour représenter les diverses fonctions de haut niveau : poser_sur, déplacer_vers, modifier_taille, ... ;

- pour générer et contrôler les démons : au niveau des boîtes englobantes, du poids, des centres de gravité, ..., à la suite de la modification d'un objet éventuellement lié par des contraintes à d'autres objets ;

- pour exécuter les différents tests ou vérifications : l'objet que l'on veut poser n'est-il pas trop lourd, ne va-t-il pas toucher quelque chose, ... ;

- pour visualiser sous X-Window System version 11 et interpréter au niveau de la couche KOOL les diverses manipulations ou cliquages.

Notre développement est réalisé sur une station DPX1000 de BULL. La version utilisée de KOOL est la toute récente version graphique KOOL V2. Nous ne disposons de cette version, et de toute sa documentation, que depuis peu ; aussi dans notre première version d'interface, tous les outils de développement et notamment les outils de dialogue personnalisés sont peu employés.

5 Résultats et conclusion

5.1 Résultats

Notre étude de faisabilité s'avère concluante puisque notre première version d'interface permet d'atteindre en partie nos objectifs. L'aide du système est présente et assez importante. Les fonctions de haut niveau sont certes assez limitées en ce qui concerne les fonctions de contrainte, mais elles nous permettent d'envisager un nombre important d'extensions :

• au niveau de la fonction *poserSur* :

- nos objets ne sont soumis, pour l'instant, à aucune rotation ; il est prévu dans un proche avenir de donner la possibilité à l'utilisateur de faire tourner ses objets. Notre condition nécessaire et suffisante pour que l'on puisse poser sur n composantes, nous l'avons vu, n'est plus valable. Une autre méthode reste à définir ;

- notre première hypothèse limite le contactBas de l'objet à poser. Nous voudrions dans l'avenir, arriver à poser un objet dont le contact_bas est

constitué de m composantes connexes, sur un objet dont le contact_haut possède n composantes connexes ;

- nos objets ne sont le résultat que d'unions entre les primitives du modeleur. Il serait intéressant de parvenir à obtenir des objets issus d'intersection ou de différence entre objets. Deux problèmes cependant se posent :

  * comment offrir une telle possibilité sous la forme de fonctions "naturelles". L'intersection et la différence sont des opérations essentiellement mathématiques et ne correspondent en effet à aucune opération intuitive pour un non informaticien ou non mathématicien ;

  * on obtiendrait des zones de contacts non plans (voir le paragraphe 4.2.3).

Toutes ces extensions relèvent certainement de la géométrie algorithmique. Notre théorème en est un exemple. Cependant, en ce qui concerne la restriction des contacts (nous ne considérons que des contacts plans), des travaux dans le domaine de la Robotique (voir [LAUG 87]) sont encore en cours. Tout n'est pas résolu, du fait de la complexité du problème.

• par l'utilisation de nouvelles fonctions :

- les fonctions simples : couper, renverser, étirer, enfoncer,...

- les fonctions de contrainte : coller contre, aligner deux axes, coller deux arêtes, ...

L'utilisation d'autres éléments caractéristiques (axe, arête, ..., très utilisés en CAO) permettrait d'envisager un bon nombre de fonctions. La richesse de ces fonctions ainsi que leur diversité semblent en mesure de fournir un outil puissant et maniable.

• par la prise en compte d'un nouveau type de contraintes : les contraintes du style

- *rigide* : un objet peut être déformable ou pas. Nous avions, dans le paragraphe 3 sur les spécifications de notre interface, fait part de ces souhaits. Nous ne disposons pas actuellement de déformation au niveau de nos primitives de base. Aussi envisageons-nous ceci parmi les extensions possibles.

- *collé* : coller contre serait distingué de placer contre dans la mesure où deux objets collés ne se détachent pas. Tout mouvement effectué sur l'un se répercute sur l'autre. Par contre, un objet A placé contre un objet B, sur sa droite par exemple, n'est amené à déplacer B que s'il effectue un mouvement vers la gauche.

Ces deux types de contrainte sont très liés. L'exemple de la figure 27 illustre bien les problèmes que le système serait amené à résoudre dans le cas où l'utilisateur veut déplacer un objet collé, posé sur un objet déformable. Imaginons un instant que l'objet déformable ne soit pas l'objet receveur mais l'objet que l'on veut déplacer. Le résultat serait différent.

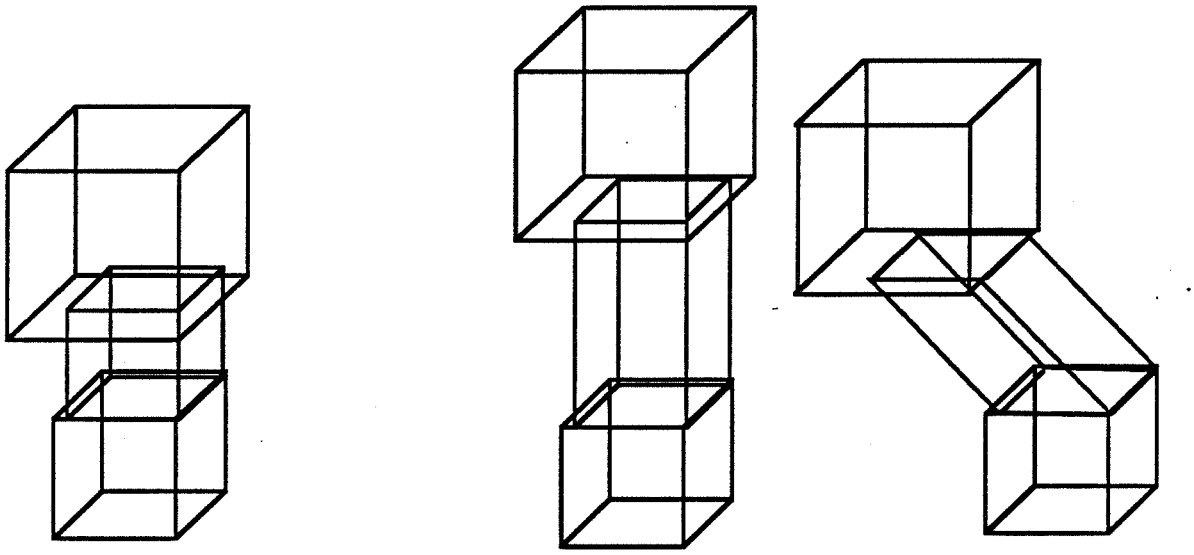


Figure 27 : Cas du déplacement d'un objet collé sur un objet déformable

- par l'introduction d'une certaine dynamique

* au niveau des contacts :

Nous proposons, pour poser un objet, une position standard. Il serait intéressant d'avoir un outil ne proposant pas toujours le même endroit pour poser. Prenons l'exemple d'une table sur laquelle sont posés plusieurs verres. Avec notre méthode, l'utilisateur se voit proposer par le système un endroit (le centre du plateau de la table) où sera posé le verre. A lui de déplacer son premier verre afin de pouvoir en poser un second.

Une méthode permettant de tenir compte de la "surface occupée", évolutive bien sûr, serait appréciable.

Nous avons signalé, dans ce manuscrit, notre impossibilité à placer interactivement, à l'heure actuelle, un objet sur deux (ou plusieurs) objets non situés à la même hauteur (voir la figure 28).

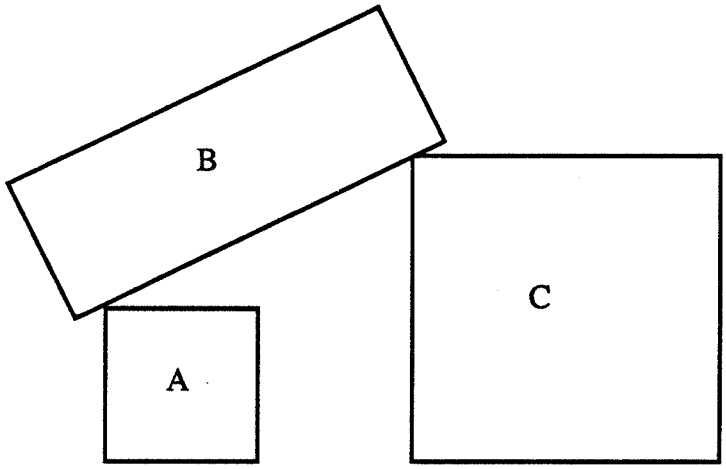


Figure 28 : Solide posé sur deux objets de hauteurs différentes

Plusieurs constatations, cependant, nous sont apparues. Toute modification de A ou de C semble remettre en cause les parties de B qui entrent en contact. Il faut donc être capable

- de déterminer le mouvement de B suite à ceux de A ou C ;

- d'en déduire alors quelles sont les parties de B qui seront amenées à toucher A et C ;

- de tenir compte de l'éventuel déséquilibre de B.

* au niveau des contraintes :
Prenons l'exemple de la figure 29. Un objet C, que l'utilisateur veut grossir, est susceptible d'entrer en contact avec un objet A.

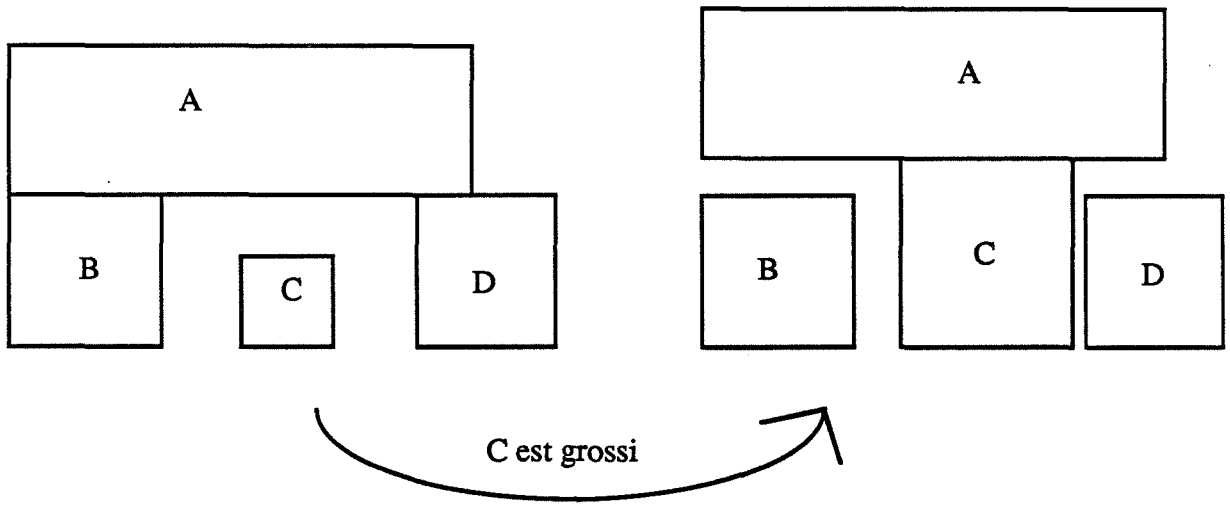


Figure 29 : Déplacement de A consécutif au grossissement de C

Dans notre interface actuelle, le système signale que l'objet C va toucher l'objet A. En cas d'acceptation de la part de l'utilisateur, l'objet B est grossi mais en aucun cas, l'objet A n'est déplacé : il n'y a pas, au moment de la demande, de relation entre ces deux objets. Il serait souhaitable de permettre une création automatique de contraintes dans ce cas de figure. Bien sûr, dans le cas où A serait collé à C ou à D, le système répondrait négativement à la demande de l'utilisateur.

5.2 Conclusion

Cette étude de faisabilité nous a également permis de tirer certaines conclusions qui vont dans le sens des remarques faites à la suite du survol des travaux existants (voir le paragraphe 2) :

- la construction par les contraintes permet de façon naturelle d'offrir un outil complet et riche pour toute conception, en particulier celle qui nous concerne, la conception d'images de synthèse ;

- une approche semi-automatique s'est imposée et s'avère être indispensable surtout en regard de toutes les extensions envisagées ;

- l'apport de l'intelligence artificielle a été bénéfique (malgré une utilisation modérée dans un premier temps) pour plusieurs raisons :

- les règles sont lisibles et la représentation des connaissances aussi ;

- l'ajout d'une règle (à la suite d'une nouvelle fonction de contrainte par exemple) est facile de par l'indépendance des règles ;

- la combinaison objets + règles permet d'organiser une méthodologie simple et puissante :

  . les règles déterminent, en fonction de la situation, quelle stratégie adopter et si besoin est, quel calcul effectuer (en affectant certaines valeurs à des attributs clés) ;

  . les démons associés aux objets génèrent ces calculs ;

• la conservation des contraintes a été traitée uniquement par le biais de messages. L'interactivité est réalisée. Nous rejoignons à ce niveau, les travaux de Braun ([BRAU 89]) qui se satisfait de l'environnement orienté objet pour résoudre ce problème, contrairement à Shonelk ([SCHO 89]) qui utilise, elle, des règles de conservation de contrainte.

CONCLUSION

Un modèle original de source de lumière en tracé de rayon a été exposé. Cette méthode, empirique, permet de concevoir diverses lampes complexes de couleur ainsi que de visualiser la partie de l'espace illuminée par la source. Elle est basée sur une représentation des sources de lumière par le biais de primitives de l'arbre de construction, ainsi que sur un algorithme de simulation de la poussière. Elle présente les avantages de ne pas générer de pré-calcul pour déterminer la portée des sources lumineuses, et de proposer une prise en compte de la poussière sans modélisation de particules. La simplicité de son utilisation, due à son intégration naturelle dans une modélisation par arbre de construction, est également à mettre à son profit. Les perspectives d'évolution de notre modèle sont nombreuses, certaines sont déjà en cours de réalisation, prenant en compte de façon plus concrète les lois de la physique. Nous pensons alors obtenir des résultats supérieurs à notre méthode.

L'étude d'une interface évoluée de CASTOR a été effectuée. Une première version d'interface a été développée. Si la voie ouverte par notre recherche nous semble prometteuse, il reste encore beaucoup d'analyses et d'expérimentations à faire en vue de permettre une conception d'images de synthèse assistée par ordinateur. Trois aspects nous semblent alors importants :

- l'apport de l'intelligence artificielle : encore minime dans notre première version, l'intelligence artificielle constitue l'indispensable outil afin de prétendre aux extensions envisagées ;

- la géométrie algorithmique : très présente en CAO ou en robotique (nous l'avons également utilisée pour obtenir une condition nécessaire et suffisante pour poser une composante sur n composantes), cette discipline pourrait peut être permettre de résoudre bon nombre des problèmes actuels ou à venir. Un approfondissement des techniques de cette discipline semble indispensable ;

- une approche semi-automatique : elle doit être constamment présente à l'esprit durant l'évolution de cette interface, sinon la combinatoire du problème dépassera rapidement la puissance du système expert développé, et interdira alors toute interactivité.

Bibliographie

[AMAR 89] M. AMARA
"Interface Graphique pour le Modeleur CASTOR"
Rapport de DEA, Ecole des Mines de Saint-Etienne, 1989.

[ARBA 89] F. ARBAB
"Examples of Geometric Reasoning in OAR"
Intel CAD System II, 1989, pp 32-57.

[ARGE 88a] J. ARGENCE
"Antialiasing for ray tracing using CSG modeling"
Proc. CG International'88, New Trends in Computer Graphics, N. Magnenat-Thalmann and D. Thalmann eds., Springer Verlag, 1988, pp 199-208.

[ARGE 88b] J. ARGENCE
"Algorithmes pour le tracé de rayons dans le cadre d'une modélisation par arbre de construction"
Thèse, Ecole des Mines de Saint-Etienne, Novembre 1988.

[BEEK 89] E. BEEKER
"Application de la programmation orientée objet à la CAO"
MICAD 89, 1989, Vol. 1, pp 121-126.

[BEIG 86] M. BEIGBEDER
"Castor : un modeleur 3D"
Actes des Journées Internationales de CAO et robotique en Architecture et BTP, Marseille, Juin 1986, pp 349-358.

[BEIG 88] M. BEIGBEDER
"Un développement pour la modélisation et la visualisation en synthèse d'images : CASTOR"
Thèse, Ecole des Mines de Saint-Etienne, avril 1988.

[BEIG 90] M. BEIGBEDER, B. PEROCHE
"Un Système de Synthèse d'Images 3D: ILLUMINES"
Rapport interne EMSE, n° 90.2, 1990.

[BERL 88] P. BERLANDIER
"Intégration d'Outils pour l'Expression et la Satisfaction de Contraintes dans un Générateur de Systèmes Experts"
Rapport de recherche I.N.R.I.A, n° 924, 1988.

[BISH 86] G. BISHOP
"Fast Phong Shading"
Computer Graphics, Vol. 20, n° 3, 1986, pp 103-106.

[BLIN 77] J.F. BLINN
"Models of light reflection for computer synthesised pictures"
Computer Or, Vol. 11, n° 2, 1977, pp 192-198.

[BLIN 78] J.F. BLINN
"Simulation of Wrinkled Surfaces"
Computer Graphics, Vol. 12, n° 3, 1978, pp 286-292.

[BROC 89] P.J. BROCK, S. COQUILLART et F. CROS
"Conception et Implémentation d'un Modeleur Géométrique Interactif"
BIGRE + GLOBULE n°61-61, 1989, pp 139-143.

[BORN 70] M. BORN and E. WOLF
"Principles of Optics"
Pergamon press, 1970.

[BORN 89] A. BORNING, M. MAHER, A. MARTINDALE and M. WILSON
"Constraint Hierarchies and Logic Programming"
Proc. of the Sixth International Logic Programming Conference, 1989,
Lisbon, pp 149-164.

[BOUA 89] K. BOUATOUCH
"Synthèse d'Image Réaliste : Algorithmes et Architectures"
Habilitation à diriger des recherches, Université de Rennes, 1989.

[BOUV 89] C. BOUVILLE and K. BOUATOUCH
"A Unified Approach to Global Illumination Models"
PIXIM 89, pp 249-263.

[BRAU 89] G. BRAUN
"Sur la Programmation de Constructions Géometriques"
PIXIM 89, pp 419-433.

[BROT 84] L.S. BROTMAN and J.B. NORMAN
"Generating soft shadows with a depth buffer algorithm"
IEEE CG&A, 1984, pp 5-12.

[CHOU 87] E. CHOURAQUI et C. INGHILTERRA
"Apport de la Méthodologie fondée sur les Objets pour la Conception d'un Système Expert d'E.A.O de la Géométrie"
MARI 87, 1987, pp 39-44.

[CHOU 90] E. CHOURAQUI, P. DUGERDIL, P. FRANCOIS, S. HANROT, A. PANCHER et P. QUINTRAND
"TECTON : un Système Intelligent de CAO en Architecture"
MICAD 90, 1990, pp 88-103.

[COHE 85] M.F. COHEN and D.P. GREENBERG
"A Progressive Refinement Approach to Fast Radiosity Image Generation"
Computer Graphics, Vol. 22, n° 4,1988, pp 75-84.

[COHE 88] M.F. COHEN and D.P. GREENBERG
"The Hemi-cube, a Radiosity Solution for Complex Environments"
Computer Graphics, Vol. 19, n° 3,1985, pp 31-40.

[COOK 82] R.L. COOK and K. TORRANCE
"A Reflectance Model for Computer Graphics"
ACM Transactions on Graphics, Vol. 1, n°1, 1982, pp 7-24.

[COOK 84] R.L. COOK
"Shade Tree"
Computer Graphics, Vol. 18, n° 3, 1984, pp 223-231.

[DAHS 89] K.E. DAHSAN and J.P. BARTHES
"Implementation Constraint Propagation in Mechanical CAD System"
Intel CAD System II, 1989, pp 217-227.

[DJED 89] N. DJEDI, R. CAUBET, M. CARCENAC, N. MATALLAH et M.Z. SANDOUK
"A Knowledge Based System for Geometric Modellng Image Synthesis"
CGI 1989, Londres, 1989, pp 399-408.

[DUFF 79] T. DUFF
"Smoothly Shaded Rendering of Pohedral Objects on raster Displays"
ACM Computer Graphics, Vol. 13, n° 2, 1979, pp 270-275.

[FERT 89a] G. FERTEY et B. PEROCHE
"Sources Directionnelles de Lumière en Tracé de Rayon"
PIXIM 89, 1989, pp 219-232.

[FERT 89b] G. FERTEY and B. PEROCHE
"Une Interface Evoluée pour un Modeleur 3D"
Actes de GroPlan 1989.

[FERT 90] G. FERTEY, B. PEROCHE and J. ZOLLER
"Creating 3D Scenes with Constraints"
Rapport interne EMSE, n° 90.4, 1990, à paraître dans les Actes de TOOLS'90.

[FLEI 87] K. FLEISHER
"Implementation of a Modeling Testbed"
Siggraph Tutorial, 1987.

[FOLE 82] J.D. FOLEY and A. VAN DAM
"Fundamentals of Interactive Computer Graphics"
the Sytems Programming Series, 1982.

[GERO 87] J.S. GERO
"Expert System in CAD"
North-Holland Amsterdam, 1987.

[GORA 84] C.M. GORAL, K.E. TORRANCE and D.P. BENNETT
"Modeling the Interaction of Light between Diffuse Surfaces"
Computer Graphics, Vol. 18, n° 3, 1984, pp 213-222.

[GOUR 71] H. GOURAUD
"Continuous Shading of Curved Surfaces"
IEEE Trans. Comput., Vol. 20, n° 6, 1971, pp 623-628.

[GRAN 86] E. GRANT, P. AMBURN, T. WHITTED
"Exploiting Classes in Modeling and Display Software"
IEEE CG&A, Nov. 86, pp 13-20.

[GREE 84] D.P. GREENBERG and C.P. VERBECK
"A Comprehensive Light Source Description for Computer Graphics"
IEEE CG&A, 1984, pp 66-75.

[GUIN 89] J.Y GUINEMER, G HUBERT, F LEBOUVIER
"La Modélisation à la Carte : les Outils Métiers"
MICAD 89, Vol. 2, 1989, pp 795-803.

[HALL 83] R.A. HALL and D.P. GREENBERG
"A testbed for realistic image synthesis"
IEEE CG&A, 1983, pp 10-20.

[HANS 90] J.M. HANSER
"CFAO et Propagation de Contraintes, Technologie, Enjeux et Etapes Industrielles"
MICAD 90, 1990, pp 202-231.

[IMME 86] D.S. IMMEL, M.F. COHEN and D.P. GREENBERG
"A Radiosity Method for Non-Diffuse Environments"
Computer Graphics, Vol 20, n° 4, 1986, pp 133-142.

[INAK 89] M. INAKAGE
"An Illumination Model for Atmospheric Environments"
Proc. of CGI' 89, 1989, pp 533-548.

[JENS 89] G.J. JENSE
"Voxel-Based Methods for CAD"
Computer-Aided Design, Vol 21, n° 8, 1989, pp 528-533.

[KAJI 86] J. T. KAJIYA
"The Rendering Equation"
Computer Graphics, Vol 20, n° 4, 1986, pp 143-150.

[KIMU 87] F. KIMURA, H. SUZUKI
"An Uniform Approach to Dimensioning and Tolerancing in Product Modeling"
Proceedings CAPE 86, 1986, pp 165-178.

[KIM 89] S.H. KIM and K. LEE
"An Assembly Modelling System for Dynamic and Kinematic Analysis"
Computer-Aided Design, Vol 21, n° 1, 1989, pp 2-12.

[KRAU 89] F.L. KRAUSE, F.H. VOSGERAU and N. YARAMANOGLU
"Implementation of Technical Rules in a Feature Based Modeller"
Intel CAD System II, 1989, pp 195-208.

[KUIJ 89] A.A.M KUIJK and E.H BLAKE
"Faster Phong Shading via Angular Interpolation"
Computer Graphics Forum 8, 1989, pp 315-324.

[LAUG 87] C. LAUGIER
"Raisonnement géométrique et Méthodes de Décision en Robotique. Application à la Programmation Automatique des Robots"
Thèse d'état, Institut National Polytechnique de Grenoble, 1987.

[LAUN 89] P. LAUNAY
"Euclid-Is Data Concept : an Object Oriented CAD/CAM System"
MICAD 89, Vol. 1, 1989, pp 173-184.

[LEE 85a] K. LEE, D. C. GOSSARD
"A Hierarchical Data Structure for Representing Assemblies : part 1"
CAD 1985, Vol. 17, n°1, 1985, pp 15-19.

[LEE 85b] K. LEE, G. ANDREWS
"Inference of the Positions of Components in an Assembly : part 2"
Computer-Aided Design, Vol 17, n° 1, 1985, pp 20-24.

[LIN 81] V.C. LIN, D.C. GOSSARD and R.A. LIGHT
"Variational Geometry in Computer-Aided Design"
Computer Graphics, Vol 15, n° 3, 1981, pp 171-178.

[LUCA 89] M. LUCAS, D. MARTIN, P. MARTIN et D. PLEMENOS
"Le projet ExploFormes, quelques pas vers la Modélisation Déclarative de Formes"
Actes de GroPlan 1989.

[MACU 90] R. MACULET et H. BOCCON-GIBOD
"ARCHIPEL : un Système d'Aide à la Conception en Architecture Industrielle"
MICAD 90, 1990, pp 106-127.

[MART 88] P. MARTIN, D. MARTIN
"An Expert System for Polyhedra Modelling"
EUROGRAPHICS ' 88, 1988, pp 221-232.

[MEYE 88] G. MEYER
"Wavelength Selection for Synthesis Image Generation"
Computer Vision, Graphics, and Image Processing, Vol. 41, n° 1, 1988, pp 57-79.

[MICH 89] D. MICHELUCCI
"CastorC"
Communications internes.

[NISH 83] T. NISHITA and E. NAKAMAE
"Half-Tone Representation of 3D Objects Illuminated by Area Sources or Polyedron Sources"
IEEE, Proc on COMPSAC, 1983, pp 237-242.

[NISH 85] T. NISHITA and E. NAKAMAE
"Shading models for points and linear sources"
ACM Trans. on Graphics, Vol. 4, n° 2, 1985, pp 124-146.

[NISH 87] T. NISHITA and E. NAKAMAE
"A Shading Model for Atmospheric Scattering considering Luminous Intensity Distribution of Light Sources"
Computer Graphics, Vol. 21, n° 4, 1987, pp 303-310.

[OHSU 89] S. OHSUGA
"Toward Intelligent CAD System"
Computer-Aided Design, Vol 21, n° 5, 1989, pp 315-337.

[PERO 88] B. PEROCHE, J. ARGENCE, D. GHAZANFARPOUR et D. MICHELUCCI
"La Synthèse d'Images"
Editions Hermès, 1988.

[PHON 75] B.T. PHONG
"Illumination for Computer Generated Pictures"
Communication of the ACM, Vol. 18, n° 6, 1975, pp 311-317.

[PLEM 89] D. PLEMENOS
"La Modélisation Déclarative de Scènes"
Rapport de recherche LIST 89-04, Nov 1989.

[ROCH 87] D. N. ROCHELEAU, K. LEE
"System for Interactive Assembly Modeling"
CAD 1987, Vol. 19, n° 2, 1987, pp 65-72.

[ROEL 90] M. ROELENS, G. FERTEY and B. PEROCHE
"Light Sources in a Ray Tracing Environment"
Rapport interne EMSE, n° 90.6, 1990.

[SABE 87] P. SABELLA, I. CARLBOM
"An Object-Oriented Approach to Solid Modeling for Empirical Data"
Siggraph Tutorial, 1987.

[SCHO 89] F. SCHONEK and A. VERROUST
"A Rule Oriented Method to Parametrize CAD Designs"
Rapport de recherche du LIENS, 1989, E.N.S.

[SHAO 88] M.Z. SHAO, Q.S. PENG and Y.D. LIANG
"A new Radiosity Approach by Procedural Refinements for Realistic Image Synthesis"
Computer Graphics, Vol 22, n° 4, 1988, pp 93- 101.

[SHEU 88] P. C-Y. SHEU
"Object Oriented Graphics Knowledge Bases"
C&G 1988, Vol. 12, n°1, 1988, pp 115-123.

[SILL 87] O. DEVILLERS, C. PUECH et F. SILLION
"CIL : un Modèle d'Illumination Intégrant les Réflexions Diffuse et Spéculaire"
Rapport de recherche du LIENS, 1987, E.N.S.

[SMIT 78] A.R. SMITH
"Color Gamut Transform Pairs"
Computer Graphics, Vol 12, n° 3, 1978, pp 12-19.

[SMIT 89] T. SMITHERS
"AI-Based Design Versus Geometry-Based Design or Why Design cannot be supported by Geometry alone"
Computer-Aided Design, Vol 21, n° 3, 1989, pp 141-150.

[SUND 87] G. SUNDE
"A CAD System with Declarative Specification of Shape"
Eurographics Worshop on Intelligent CAD, Noordwijkerhout, Avril 1987.

[TORR 67] K. TORRANCE and E. SPARROW
"Theory for Off-Specular Reflection from Roughened Surfaces"
J. Opt. Soc. Am., Vol. 57, n° 9, 1967, pp 1105-1114.

[VAN 88] M.J.G.M. VAN EMMERIK
"A System for Graphical Interaction on Parametrized Models"
EUROGRAPHICS '88, 1988, pp 233-242.

[WARN 69] J.E. WARNOCK
"A Hidden Surface Algorithm for Computer Generated Half-tone Pictures"
University of Utah, Tr 4-5, C5 Department, 1969.

[WARN 83] D.R. WARN
"Lighting Controls for Synthetic Images"
Computer Graphics, Vol. 17, n° 3, 1983, pp 13-21.

[WHIT 80] T. WHITTED
"An Improved Illumination Model for Shaded Display"
Communications of the ACM, Vol. 23 (6), 1980, pp 343-349.

[WALL 87] J.R. WALLACE, M.F. COHEN and D.P. GREENBERG
"A two-pass Solution to the Rendering Equation : a Synthesis of Ray Tracing and Radiosity Methods"
Proc SIGGRAPH' 87, Computer Graphics, Vol 21, n° 4, 1987, pp 311- 320.

ECOLE NATIONALE SUPERIEURE
DES MINES DE SAINT-ETIENNE

N° d'ordre : 40 ID
1990

Gilles FERTEY

Deux problèmes en synthèse d'images :

les sources directionnelles de lumière et une interface évoluée.

Mots clés :

Synthèse d'images, tracé de rayon, arbre de construction, modèles d'éclairement, modèles de sources de lumière.

Intelligence artificielle, programmation orientée vers les objets, système expert, CAO, construction par contrainte.

Résumé :

Le présent travail est intégré au sein du logiciel interne de synthèse d'images, ILLUMINES. Il porte sur deux points :

Les lampes de bureau, les phares de voitures, les lampadaires dans les rues,... éclairent le plus souvent selon une direction et une portée déterminées. Nous présentons dans la première partie de la thèse une méthode permettant de simuler ce type de source lumineuse, avec une modélisation par arbre de construction, dans un environnement de tracé de rayons. Notre méthode permet de visualiser les zones éclairées par le faisceau de la source lumineuse, mais également le faisceau lui-même.

La conception d'une image de synthèse s'avère, pour un non-spécialiste, une réalisation pénible et difficile. A partir d'une hiérarchie orientée objet, d'un moteur d'inférence et des techniques interactives traditionnelles, nous proposons à l'utilisateur un outil d'aide à la conception d'images de synthèse simple et efficace. Une construction par les contraintes est générée par le moteur d'inférence et rendue accessible par l'utilisation de la fonction de contrainte : poser_sur. Diverses autres fonctions de haut niveau sont proposées pour la modification de la scène : déplacer_vers, modifier_taille, éloigner_de, rapprocher_de, ... La préservation des contraintes est effectuée par envoi de messages. Ce prototype d'interface évolué interactif a été implanté au sein du logiciel ILLUMINES et permet d'envisager ce qui est en droit de s'appeler, la conception d'images de synthèse assistée par ordinateur.