UNIVERSITE DE LIMOGES

ECOLE DOCTORALE Science – Technologie – Santé

FACULTE des Sciences et Techniques

Année : 2004 **Thèse N° [ ]**

Thèse

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITE DE LIMOGES

Discipline / Spécialité : Informatique

présentée et soutenue par

M. Patrick POULINGEAS

le 18 novembre 2004.

SPECIFICATION DECLARATIVE

DE L'AMBIANCE D'UNE SCENE

Thèse dirigée par M. Dimitri PLEMENOS

JURY :

Professeur René Caubet, Université de Toulouse	**Président et Rapporteur.**
Professeur Michel Mériaux, Université de Poitiers	**Rapporteur.**
Professeur Djamchid Ghazanfarpour, Université de Limoges	**Examinateur.**
Professeur Dimitri Plemenos, Université de Limoges	**Examinateur.**
M. Vincent Jolivet (Maître de conférences), Université de Limoges	**Examinateur.**

Remerciements.

Je tiens à exprimer toute ma reconnaissance à MM. Plemenos et Jolivet pour leur encadrement, leurs nombreux conseils et leur soutien constant tout au long de ma thèse.

M. Caubet m'a fait l'honneur d'accepter d'être président de mon jury de thèse et rapporteur de celle-ci. Pour cela, ainsi que pour ses commentaires sur mon mémoire, je lui exprime ma profonde gratitude.

Je remercie M. Mériaux d'avoir accepté d'être rapporteur de ma thèse, ainsi que pour ses jugements très pertinents sur mon manuscrit, tant sur le fond que sur la forme.

Je remercie M. Ghazanfarpour pour avoir accepté de faire partie de mon jury de thèse. J'ai beaucoup appris en infographie grâce à son livre écrit en collaboration avec M. Péroche.

Je remercie tous les chercheurs, enseignants et membres du personnel du laboratoire MSI pour leur amitié et leur aide pendant ces trois années de thèse. Notamment notre toujours jeune et pétillante Suzy, l'auguste Horace qui seul goûte à Jidé mes extravagants développements borgésiens et postmodernes, Sylvain le relieur admirable et Kevin l'Excellence de l'Excellence.

Je tiens à témoigner tout particulièrement ma sympathie et ma reconnaissance à Hubert pour m'avoir supporté dans son bureau durant ces dernières années, et ce avec un stoïcisme inégalable. De plus, grâce à lui, j'aurai pu découvrir quelques mystères de l'administration système. Il fait incontestablement partie des gens sans qui ce travail n'aurait sans doute jamais abouti.

Je remercie Serge Bailly et Cédric Peyronnet pour leur accueil chaleureux quand j'ai eu à (modestement) intervenir en TIC.

J'inclus ici une liste d'amis (rangée par ordre alphabétique), qui, ces dernières années, ont beaucoup compté pour moi :

Bertrand Bach, Véronique et Christophe Cadic, Leïla et Denis Carabin, Stéphane Dellière, Sylvia Estivie, Chantal et Pierre Fourest, Sylvain Loulergue, Frédéric Royer, Jacques Texier, Frédéric Terracher et Emmanuel Weppe.

Si Marcel Proust ne croyait pas en l'amitié, c'est parce qu'il n'a pas connu les miens.

Il m'est bien évidemment impossible de ne pas citer l'Homme qui a été la plus grande chance de ma vie : Friedrich Nietzsche. Ma rencontre avec les écrits incomparables de ce philosophe est pour moi d'une importance si grande qu'il est indécent de parler simplement de dette. Qui est d'ailleurs parvenu aux hauteurs de Zarathoustra pour en parler avec la profondeur et la joie qui conviennent ?

Je termine par un grand remerciement à mes parents auxquels je dédie ce mémoire.

« Jamais je n'avais vécu pareil automne, ni cru que chose semblable fût possible sur terre – un Claude Lorrain prolongé en esprit à l'infini, chaque jour de la même irrépressible perfection. »

(Friedrich Nietzsche. Ecce Homo. 1888**)**

TABLE DES FIGURES.

Partie I : Introduction

1. Présentation.

Depuis ces dernières années, les modeleurs mis à la disposition tant du grand public que des professionnels de l'infographie ont vu leur fonctionnalités augmenter quantitativement de manière impressionnante. Il suffit de comparer le nombre de barres d'outils et d'icônes apparaissant à l'écran quand on lance aujourd'hui un modeleur avec la sobriété des interfaces des modeleurs du début des années 80 (sobriété qui caractérisait une pauvreté des outils proposés au concepteur et non une volonté délibérée de simplifier l'interface homme-machine du logiciel). L'accroissement des capacités des modeleurs s'est toutefois accompli sur les bases conçues dès les premiers temps de l'informatique graphique. Les nouveaux modeleurs s'appuient encore sur une approche géométrique de bas niveau. De façon paradoxale, la prise en main d'un modeleur est devenue de nos jours peut-être plus délicate qu'autrefois : la multiplication des fonctionnalités aboutit à une complexité qui n'est compensée par aucun outil d'aide à la conception qui permettrait une approche synthétique de la construction d'une scène. L'accumulation d'innovations simplement techniques n'est vraiment ressentie positivement que par ceux qui ont pu suivre l'évolution des modeleurs au cours des ans. Sortie de son histoire, cette prolifération d'outils n'est pas loin de perdre son sens. S'il est encore possible pour un graphiste non familiarisé avec ces modeleurs de parvenir à une maîtrise de ces logiciels, il n'est pas évident qu'un non spécialiste arrive à exploiter même de façon élémentaire de tels modeleurs où l'unité des outils n'existe pratiquement plus. L'absence d'outils d'aide à la conception de haut niveau sera de toute manière ressentie négativement par ces deux catégories d'utilisateurs (spécialiste ou novice).

La modélisation déclarative ([LD95], [GAI03]) a essayé de remédier à cette situation. A des notions exclusivement géométriques et numériques est substitué le concept de propriété. Les caractéristiques principales d'une scène peuvent en effet se décrire à l'aide d'un ensemble de propriétés exprimées dans un langage proche d'une langue ou à l'aide d'une interface graphique permettant des spécifications gestuelles. Une propriété correspond à une somme de manipulations élémentaires dans un modeleur géométrique usuel. Remarquons qu'il n'y a pas de correspondance bi-univoque entre une propriété de haut niveau et des valeurs précises pour diverses fonctionnalités élémentaires d'un modeleur traditionnel. Plusieurs combinaisons de valeurs pour les paramètres de certaines fonctionnalités élémentaires peuvent être associées à une propriété. Ainsi, plusieurs scènes seront produites pour une même propriété. Parlons d'ailleurs plutôt d'ébauches de scènes. Les propriétés mises à la disposition de l'utilisateur d'un modeleur déclaratif ne permettent pas de spécifier des détails extrêmement fins d'une scène. Remarquons que ce n'est pas leur rôle : les propriétés sont liées à des concepts de haut niveau qui relèvent de la structure globale d'une scène. Ces concepts peuvent être des notions comme celles d'occupation spatiale (dans une boîte englobante), de position relative entre deux éléments de la scène, de décomposition hiérarchique d'un élément en plusieurs constituants (ce qui renvoie à une démarche de conception analytique pour la construction d'une scène). Les concepts que nous avons évoqués ont ainsi été mis en œuvre dans le modeleur déclaratif MultiFormes ([PLE95]).

En résumé, à partir de propriétés de haut niveau décrivant la structure d'une scène, un modeleur déclaratif engendre une suite d'esquisses de scènes satisfaisant ces propriétés. L'utilisateur pourra ensuite terminer la fabrication de sa scène dans un modeleur géométrique qui lui offrira des outils permettant de travailler les détails de sa scène.

On distingue habituellement deux types de modeleurs déclaratifs : les modeleurs déclaratifs généralistes et les modeleurs déclaratifs spécialisés. Les premiers prennent en

charge l'aide à la conception d'une scène quelconque. Les propriétés manipulées ressemblent alors à celles que nous avons indiquées en exemple dans le paragraphe précédent (MultiFormes étant un modeleur déclaratif généraliste). Les modeleurs déclaratifs spécialisés, quant à eux, se focalisent sur un domaine précis (génération de polyèdres, de mégalithes, de stalactites, etc.).

Nos recherches se placent dans un champ proche de celui des modeleurs déclaratifs spécialisés. A partir d'une géométrie fixée d'une scène, nous cherchons à procurer à l'utilisateur des moyens simples à mettre en œuvre afin qu'il puisse introduire une certaine ambiance dans cette scène. Certains logiciels permettant de créer une ambiance existent bien entendu déjà. Par exemple, Terragen™ ([TER04]) est un outil pour la fabrication d'images très réalistes de paysages. Il peut gérer des conditions atmosphériques complexes (aspect du ciel, nuages, brouillard, etc.). Mais ceci s'accomplit en choisissant de nombreux paramètres numériques dont l'influence sur l'ambiance globale est parfois difficile à appréhender. Nous avons donc une analogie entre ce logiciel et les modeleurs géométriques courants : ils obligent le concepteur à se concentrer sur des données de bas niveau et la détermination des bonnes valeurs pour les paramètres peut n'intervenir qu'au bout d'un processus d'essais-corrections extrêmement long.

2. Motivations de nos recherches.

Les travaux que nous avons entrepris ont eu pour but de faciliter la création d'une ambiance pour une scène en mettant à la disposition du concepteur des propriétés sous la forme d'un sous-ensemble d'une langue (le français en l'occurrence).

Nous avons examiné la pertinence de certaines méthodes employées usuellement en modélisation déclarative (Modélisation des propriétés sous forme de sous-ensembles flous, recours à des CSP).

Plutôt que d'essayer de gérer de façon exhaustive les propriétés en relation avec l'ambiance d'une scène, nous avons cherché à dégager une méthodologie générale à même de prendre en compte tous les concepts offerts par le modèle d'éclairement du logiciel de rendu utilisé en fin du processus de création.

Dans un premier temps, afin de restreindre le problème, nous nous sommes limités à l'ambiance dans un paysage. Dans le cadre général de l'ambiance d'une scène, nous n'avons considéré que le problème de l'éclairage (appelé éclairage inverse puisqu'il s'agit de trouver les paramètres d'une ou plusieurs sources de lumière qui seront chargées de produire l'éclairage souhaité par l'utilisateur). Ces limitations, rendues nécessaires par l'ampleur du problème et la variété des domaines intervenant, n'ont toutefois aucune influence, ni sur la capacité de généralisation de la méthode que nous avons développée, ni sur les analyses que nous avons menées au cours de nos travaux.

3. Apports de nos recherches.

Appuyant notre modélisation des propriétés sur la théorie des sous-ensembles flous, nous avons prolongé les travaux de [DES95] en introduisant un nouveau type de spécifications : des descriptions contenant une incertitude quant à la propriété intervenant. Par exemple, nous permettons au concepteur d'utiliser des spécifications comme : « Il est *plutôt certain* que la table est très éclairée. ». Nous mettons ainsi à la disposition de l'utilisateur un ensemble d'opérateurs l'autorisant à indiquer de façon naturelle le degré d'incertitude qu'il veut donner (ou non) à une description d'une caractéristique de l'ambiance. Nous avons intégré la gestion de ces opérateurs d'incertitude dans le cadre de la théorie des sous-ensembles flous (Ils ne sont donc pas traités de façon différentes des autres opérateurs influant sur les propriétés de base).

Pour l'éclairage inverse, nous avons développé une nouvelle méthode dans le cas de l'éclairage inverse direct. Cet algorithme repose sur une méthode de Monte-Carlo et s'applique avec le modèle d'éclairement de la radiosité.

En essayant d'intégrer notre méthode d'éclairage inverse dans un système global de spécification déclarative de l'ambiance d'une scène similaire à ceux existant en modélisation déclarative, nous avons constaté la difficulté de simplement reprendre les diverses techniques de la modélisation déclarative. De l'analyse des difficultés rencontrées, nous avons dégagé le cœur de notre problématique : le représentation et la gestion d'un ensemble très grand de scènes solutions. Nous nous sommes ainsi recentrés sur des questions relevant de la visualisation de l'information. Nous avons alors proposé une adaptation des méthodes généralistes de [MAR97] en introduisant une construction spécifique d'un graphe des solutions. Ce graphe est fortement structuré, avec des premiers niveaux sous forme d'arborescence ce qui permet un classement plus pertinent que ceux présentés dans [MAR97]. Les derniers niveaux de notre graphe des solutions ont recours à une notion de distance entre deux scènes. Nous avons défini cette distance de telle sorte que soit mesurée la différence entre l'impact lumineux d'une source de lumière sur les deux scènes.

L'une des conséquences majeures de nos recherches est que nous avons été amenés à mieux cerner les contraintes d'une condition de départ de nos travaux : le fait que la géométrie de la scène ne change pas. La création d'une ambiance particulière est en fait le résultat d'un processus d'ajustement virtuellement sans fin entre la géométrie et les autres paramètres d'une scène (comme les caractéristiques physiques d'une source lumineuse). Il y a en effet une ambiance propre aux objets placés dans la scène (donc à la géométrie de la scène). Aussi, tenter d'imposer une ambiance globale en opposition avec l'ambiance se dégageant de la géométrie même n'aboutit qu'à des scènes sémantiquement incohérentes. On ne peut qu'affaiblir ou renforcer l'ambiance intrinsèque aux objets si l'on choisit de ne point modifier la géométrie d'une scène. Remarquons que la vérification de la compatibilité d'un objectif d'ambiance global avec l'ambiance portée par les divers éléments de la scène est un problème extrêmement complexe que nous n'avons pas abordé.

4. Structure du mémoire.

Dans la deuxième partie de ce mémoire, nous allons présenter notre étude sur l'ambiance dans un paysage. Ceci nous permettra de développer notre modélisation des propriétés à l'aide de sous-ensembles flous. Nous illustrerons ensuite la pertinence de la représentation des propriétés que nous avons choisie avec deux propriétés : l'aspect brumeux et l'éclairage naturel d'un paysage. Nous avons employé comme format de données pour nos scènes le standard VRML [VRML97]. Ce format textuel permet de structurer aisément une scène et de modifier facilement les paramètres des éléments de cette scène.

La troisième partie est consacrée à l'ambiance dans une scène quelconque et plus spécifiquement à l'éclairage inverse.

Après quelques réflexions sur la notion d'ambiance, nous avons fait un état de l'art sur l'éclairage inverse accompagné de remarques sur des domaines apparentés. Puis nous détaillerons une nouvelle méthode d'éclairage inverse recourant à un procédé de Monte-Carlo. Nous examinerons par la suite la possibilité de coupler cette méthode avec une utilisation des CSP (technique souvent employée en modélisation déclarative). Nous verrons que les CSP obtenus seront d'une nature très particulière et nécessiteront une procédure de résolution spécifique. Le problème qui apparaîtra enfin sera la gestion d'un ensemble très élevé de solutions pour une spécification déclarative.

Afin de présenter de manière adéquate l'ensemble des scènes solutions, nous reporterons notre attention sur la visualisation de l'information. Nous étudierons principalement le

paradigme des *Design Galleries* exposé dans [MAR97]. Nous proposerons ensuite une méthode de classement des scènes solutions sous forme de graphe. Ce graphe sera produit en s'aidant des connaissances que nous avons sur notre contexte de travail (Eclairage inverse, radiosité). Ces connaissances permettront d'obtenir une meilleure classification des solutions comparativement aux *Design Galleries* qui sont de nature généralistes.

La partie 4 constituera la conclusion de notre mémoire. Nous reviendrons sur les travaux effectués et repositionnerons notre outil pour la création d'une ambiance en tenant compte de nos remarques sur l'ambiance se dégageant a priori des objets de la scène. Nous indiquerons finalement diverses améliorations et des axes de recherche que nous envisageons pour le futur.

Partie II : Spécification déclarative de l'ambiance pour un paysage.

1 Introduction.

Avant d'étudier le problème de la création d'une ambiance dans une scène quelconque dans la partie III, nous allons examiner un cas particulier : celui de l'ambiance dans un paysage.

Les propriétés que nous allons considérer sont l'éclairage naturel et l'aspect brumeux d'une scène. Il va de soi que beaucoup d'autres propriétés sont à même d'influer sur l'ambiance d'un paysage. Notre objectif n'est cependant pas de traiter de façon exhaustive les différents concepts contribuant à l'ambiance d'un paysage, mais plutôt de développer une méthodologie pour gérer des spécifications déclaratives se rapportant à des propriétés de haut niveau d'une scène. Cette méthodologie obtenue, on pourra prendre en compte de nouvelles propriétés pour produire une ambiance particulière dans un paysage, dans la mesure où l'on disposera d'un logiciel de rendu suffisamment puissant pour gérer efficacement les paramètres physiques liés à ces nouvelles propriétés.

Dans le chapitre suivant, nous allons donc introduire un moyen de modéliser les propriétés apparaissant dans une description déclarative : la théorie des sous-ensembles flous. Nous présenterons ensuite une application de cette technique de modélisation à la spécification déclarative de l'ambiance d'un paysage. Ceci nous permettra d'examiner la pertinence de notre approche. Nous analyserons les résultats obtenus et soulignerons notamment que notre modélisation à base de sous-ensembles flous, par sa nature généraliste, pourra s'étendre à la gestion de l'ambiance d'une scène quelconque.

2 Représentation de propriétés à l'aide de sous-ensembles flous.

2.1 Introduction.

Quand nous décrivons une propriété d'une scène comme « la table est grande », nous introduisons une certaine imprécision dans notre description. Quand nous passons à une formulation numérique de la propriété, il n'y a pas de raison pour que nous introduisions une longueur minimale en dessous de laquelle on considèrerait que la table n'est pas grande et au-dessus de laquelle on estimerait au contraire que la table est grande. Plutôt que d'avoir une valeur seuil qui sépare brutalement l'ensemble des tables possibles entre les tables grandes et celles qui ne le sont pas, il paraît plus naturel d'avoir un nombre qui estime le degré de satisfaction de la propriété. Pour la propriété que nous avons prise en exemple, ce nombre diminuerait progressivement quand la longueur de la table diminuerait aussi.

2.2 La notion de sous-ensemble flou.

Pour prendre en compte l'imprécision due à la nature déclarative des spécifications, nous proposons de recourir au concept de sous-ensemble flou [GAC97].

A partir d'un ensemble de référence E, on définit un sous-ensemble flou A par une fonction d'appartenance $\mu_A : E \rightarrow [0,1]$. Pour $x \in E$, $\mu_A(x)$ mesure le degré d'appartenance de x au sous-ensemble flou A. Il est d'usage de confondre le sous-ensemble flou et sa fonction d'appartenance quand il n'y a pas d'ambiguïté.

Nous allons voir que seules certaines formes de fonctions d'appartenance vont nous intéresser pour nos travaux.

On introduit par ailleurs les définitions suivantes :

- Noyau(A) = $\{x \in E, \mu_A(x) = 1\}$
- Support(A) = $\{x \in E, \mu_A(x) \neq 0\}$

Un exemple de sous-ensemble flou apparaît en figure 1.

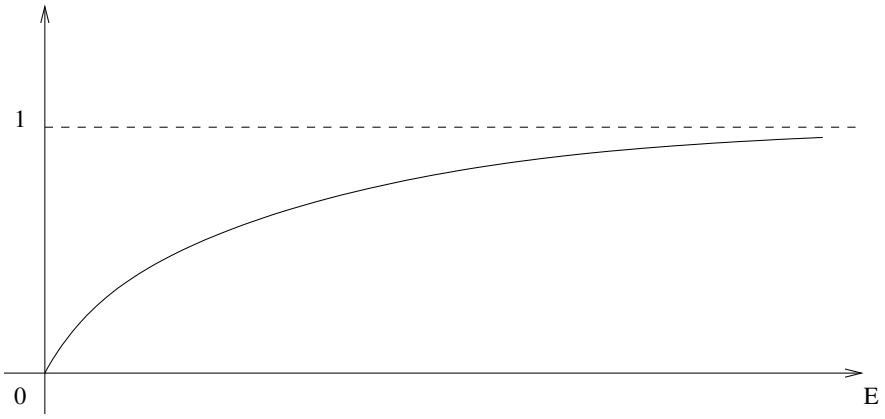


Figure 1. Exemple de sous-ensemble flou.

Pour le sous-ensemble flou A de la figure 1, on a :

- Noyau(A) = ∅
- Support(A) = E \ {0}

La notion de propriété se nomme en théorie des sous-ensembles flous un prédicat unaire. Un prédicat unaire n'est rien d'autre qu'un sous-ensemble flou.

Dans l'exemple présenté en introduction, on pourrait considérer la propriété « grand » comme un prédicat unaire. Pour la propriété « grand », les x ∈ Noyau(« grand ») seront considérés comme « absolument » grands, alors que les valeurs de l'ensemble Support(« grand ») seront « plus ou moins » grandes.

Pour modéliser un ensemble de propriétés de même nature, comme {« très grand », « grand », « moyen », « petit », « très petit »}, on utilise en théorie des sous-ensembles flous une famille de prédicats. Nous verrons que grâce à l'utilisation de modificateurs, nous pourrons calculer les fonctions d'appartenance de plusieurs propriétés à l'aide simplement d'une propriété de base (voire d'un ensemble restreint de propriétés de base).

2.3 Les différents types de propriétés et leurs modificateurs.

Une première taxinomie des différents types de propriétés que l'on peut utiliser pour une spécification déclarative des souhaits du concepteur a été réalisée en modélisation déclarative par [DES95]. Dans le cadre de ses travaux, l'ensemble de référence est appelé domaine.

2.3.1 Les propriétés simples.

Une propriété simple représente un aspect qualitatif d'une scène ou d'une partie d'une scène. La description : « la table est grande » met ainsi en œuvre la propriété simple « grand ».

Certaines propriétés simples peuvent accepter des modificateurs (comme « très », « peu », « faiblement », etc.). Ce sont ces propriétés qui vont intervenir dans nos travaux.

Pour modéliser une propriété simple, [DES95] propose d'utiliser un sous-ensemble flou avec une fonction d'appartenance particulière, de type L-R. Ces fonctions, définies sur un domaine D, se caractérisent par un quadruplet (α,a,b,β) et deux fonctions L et R appelées fonctions de forme qui sont décroissantes et semi-continues supérieurement. La fonction

d'appartenance f associée à un quadruplet (α,a,b,β) et à deux fonctions de forme L et R est définie par la formule suivante :

$$f : D \to [0,1]$$

$$t \mapsto \begin{cases} 0 & si\ t < a - \alpha \\ L\left(\dfrac{a-t}{\alpha}\right) & si\ \alpha \neq 0\ et\ (a - \alpha \leq t < a) \\ 1 & si\ a \leq t \leq b \\ R\left(\dfrac{t-b}{\beta}\right) & si\ \beta \neq 0\ et\ (b < t \leq b + \beta) \\ 0 & si\ t > b + \beta \end{cases}$$

On a des propriétés intéressantes pour ce genre de fonctions d'appartenance :
- Noyau(f) = [a,b],
- Support(f) =]a-α,b+β[,
- ((α>0 ou β>0) et a≠b) ⇔ Le sous-ensemble flou associé à f est un intervalle flou [GAC97]

Un exemple de fonction d'appartenance de type L-R est donné en figure 2.

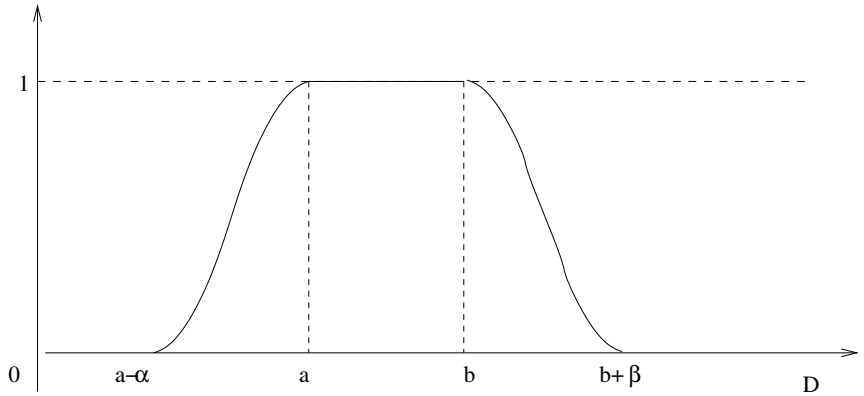


Figure 2. Exemple de fonction d'appartenance de type L-R.

2.3.2 Les modificateurs d'une propriété simple.

Plutôt que d'avoir une famille de prédicats pour représenter un ensemble de propriétés comme : {« très froid», «assez froid», « froid », « assez peu froid », « très peu froid »}, il est préférable de n'avoir qu'à spécifier des paramètres pour la propriété de base « froid » et de disposer de formules permettant de calculer si besoin la fonction d'appartenance liée à une propriété sur laquelle est appliqué un modificateur (comme « très », « assez », etc.).

Pour un concept donné, il suffira généralement de donner une ou trois propriétés de base. Ainsi pour le concept de température, les trois propriétés « froid », « moyen » et « chaud » permettront à elles seules d'obtenir toute une série de propriétés grâce à l'application de modificateurs sur celles-ci. Il est possible de déterminer de façon automatique les paramètres caractéristiques d'une propriété de base à partir d'un domaine borné, avec par exemple les formules de [DES98].

Dans [DES95] et [DP97] sont développées des techniques pour déterminer la fonction d'appartenance f' qui est associée à une propriété du genre : « m P » où m est un modificateur

et P une propriété simple. La fonction f' dépend de la fonction d'appartenance f liée à P et du modificateur m, mais aussi d'autres paramètres issus de la sémantique de la propriété P.

En pratique, le résultat de l'application d'un modificateur aboutit à une translation et à une contraction (ou à une dilatation) pour la fonction d'appartenance de la propriété de départ.

Pour des raisons d'homogénéité dans l'écriture des propriétés, on peut considérer qu'un modificateur s'applique systématique sur une propriété simple : le modificateur vide (« ∅ ») qui ne change aucunement la fonction d'appartenance associée à la propriété.

Avec la richesse d'une langue, on pourrait introduire un nombre incalculable de modificateurs, aussi pour des raisons pratiques, nous avons choisi de nous limiter à l'ensemble des modificateurs suivants : {« extrêmement », « très », « assez », « ∅ », « assez peu », « très peu », « extrêmement peu »}.

Utiliser un modificateur revient à translater et à contracter (ou dilater) la fonction d'appartenance. Il faut cependant noter que la direction de la translation est dépendante de la sémantique de la propriété. En effet, le modificateur « très » va translater dans des directions opposées les fonctions d'appartenance des propriétés « chaud » et « froid ». Les propriétés qui sont translatées dans le sens positif pour un modificateur comme « très » (par exemple : « chaud ») sont qualifiées de positives. Inversement, « froid » est une propriété négative. Quand aux propriétés qui ne sont pas translatées (comme « moyen »), elles sont appelées propriétés neutres.

Ainsi, dans notre approche, une propriété n'est pas seulement modélisée par un sous-ensemble flou. D'autres paramètres, liés à la sémantique de la propriété, entrent en compte pour l'exploitation d'une propriété dans une spécification déclarative d'un objectif à atteindre.

En plus du sens de la translation, l'importance de cette translation dépendra du modificateur mais aussi de la sémantique de la propriété. Pour gérer ceci, nous allons donc associer un nombre réel t, appelé coefficient de translation, à chaque propriété.

Le coefficient de translation t d'une propriété sera tel que :

- $t > 0$ pour les propriétés positives,
- $t = 0$ pour les propriétés neutres,
- $t < 0$ pour les propriétés négatives.

Par ailleurs, certains modificateurs vont accentuer (« très ») ou diminuer (« très peu ») la propriété sur laquelle ils agissent. Pour gérer ceci, nous allons associer un coefficient de modification k à chaque modificateur. Dans les applications de notre recherche, nous avons fait les assignations suivantes :

$k_{extrêmement} = 6$
$k_{très} = 2$
$k_{assez} = 1$
$k_{\varnothing} = 0$
$k_{assez\ peu} = -1$
$k_{très\ peu} = -2$
$k_{extrêmement\ peu} = -6$

Ces choix correspondent à ceux de [DP97].

Soit une propriété P complètement déterminée par un quadruplet (α,a,b,β), deux fonctions de forme L et R et un coefficient de translation t.

Soit m un modificateur de coefficient de modification k.

La propriété « m P » a pour fonction d'appartenance f' et pour coefficient de translation t' tels que :

$$\alpha' = \alpha(1 - c|k|)$$
$$\beta' = \beta(1 - c|k|)$$
$$a' = a + kt + \frac{1}{2}|kc(b-a)|$$
$$b' = b + kt - \frac{1}{2}|kc(b-a)|$$
$$t' = signe(k).signe(t).|t|$$
$$L' = L$$
$$R' = R$$

Dans les applications, il est usuel de prendre c = 10%.

Sur la figure 3, on peut observer l'effet de modificateurs sur une propriété négative :

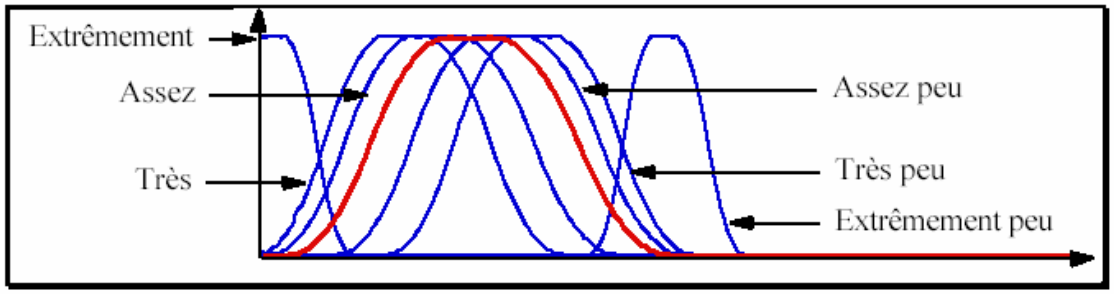


Figure 3. Application de modificateurs à une propriété simple (en rouge sur la figure – figure extraite de [DP97]).

Remarque sur la composition des modificateurs :

On ne peut pas composer n'importe quels modificateurs choisis dans l'ensemble de référence que nous avons pris par convention. Ainsi, une description comme : « La table est assez très peu grande » n'a aucun sens. Par contre, la spécification : « La table est très très grande » est tout à fait valable et utilise une propriété qui peut être modélisée avec la technique que nous venons d'exposer.

2.3.3 Les propriétés paramétrées.

Ces propriétés font intervenir dans leur expression linguistique une ou plusieurs valeurs du domaine. Par exemple :

- « La hauteur de la lampe est de 50 cm », (1)
- « La hauteur de la lampe est entre 40 et 60 cm ». (2)

Bien qu'une valeur numérique apparaisse, il y a toujours une imprécision qui intervient (imprécision peut-être plus naturelle dans la description (2) que dans la description (1)). Nous pouvons donc modéliser ces propriétés par des fonctions d'appartenance de type L-R comme nous l'avons fait pour les propriétés simples.

Dans le cas où une seule valeur v du domaine joue un rôle dans la spécification de la propriété (comme pour la description (1) précédente), on peut prendre a = b = v et la propriété

est alors représentée par un nombre flou [GAC97]. Dans le reste de ce travail, nous avons préféré choisir a = v-ε et b = v+ε où ε est déterminé par une heuristique dépendant de la nature de la propriété étudiée.

On notera que l'on ne peut pas appliquer un modificateur à une propriété paramétrée. Une spécification comme « La hauteur de la lampe est très de 50 cm » n'a en effet pas de sens. Toutefois, pour des raisons de commodité d'écriture, on peut parfois considérer que le modificateur vide s'applique par défaut à une propriété paramétrée.

2.3.4 Les opérateurs flous.

Il s'agit d'une catégorie de termes linguistiques comme « plus ou moins » ou « vaguement ». Les opérateurs flous peuvent s'appliquer à des propriétés simples ou à des propriétés paramétrées.

Comme pour les modificateurs, nous pourrions en trouver une grande quantité dans une langue. Nous nous limiterons à l'ensemble suivant pour nos applications : {« exactement », « vraiment », « ∅ », « plus ou moins », « environ », « vaguement »}. De façon similaire à ce qui se passait pour les modificateurs, on peut considérer que l'opérateur flou « ∅ » s'applique à une propriété simple ou paramétrée de base.

L'application d'un opérateur flou aboutit à une dilatation ou à une contraction de la fonction d'appartenance de la propriété de base. Un opérateur flou tel que « vraiment » procède à une contraction qui correspond à une diminution de l'imprécision. Au contraire, l'opérateur « vaguement » aboutit à une dilatation correspondant à une augmentation de l'imprécision de la description.

Contrairement à ce qui a lieu avec un modificateur, il n'y a pas de translation quand on recourt à un opérateur flou. Il n'y a aucun changement du noyau de la fonction d'appartenance.

Chaque opérateur flou est caractérisé par un coefficient j qui est un réel strictement positif. On a les propriétés suivantes :

- $j = 1$ pour l'opérateur flou ∅,
- $j > 1$ pour un opérateur effectuant une contraction (comme « vraiment »),
- $0 < j < 1$ pour un opérateur effectuant une dilatation (comme « vaguement »).

Pour l'ensemble des opérateurs flous que nous avons sélectionnés, nous avons choisi les coefficients suivants :

$$j_{exactement} = 4$$
$$j_{vraiment} = 2$$
$$j_{\varnothing} = 1$$
$$j_{plus\ ou\ moins} = 0.5$$
$$j_{environ} = 0.25$$
$$j_{vaguement} = \frac{1}{6}$$

De même que pour les coefficients des modificateurs, ces valeurs correspondent à celles de [DP97].

Soit une propriété P complètement déterminée par un quadruplet (α,a,b,β), deux fonctions de forme L et R et un coefficient de translation t.

Soit o un opérateur flou de coefficient j.

La propriété «o P » a pour fonction d'appartenance f' et pour coefficient de translation t' tels que :

$$\alpha' = \begin{cases} \alpha(1 - jc) & si \quad j \in]1,+\infty[\\ \alpha & si \quad j = 1 \\ \alpha(1 + \frac{c}{j}) & si \quad j \in]0,1[\end{cases}$$

$$\beta' = \begin{cases} \beta(1 - jc) & si \quad j \in]1,+\infty[\\ \beta & si \quad j = 1 \\ \beta\left(1 + \frac{c}{j}\right) & si \quad j \in]0,1[\end{cases}$$

$$a' = a$$

$$b' = b$$

$$t' = t$$

$$L' = L^j$$

$$R' = R^j$$

Dans les applications, il est courant de prendre c = 10% comme pour les modificateurs.

Sur la figure 4, on peut voir le résultat de l'application d'opérateurs flous sur une propriété :

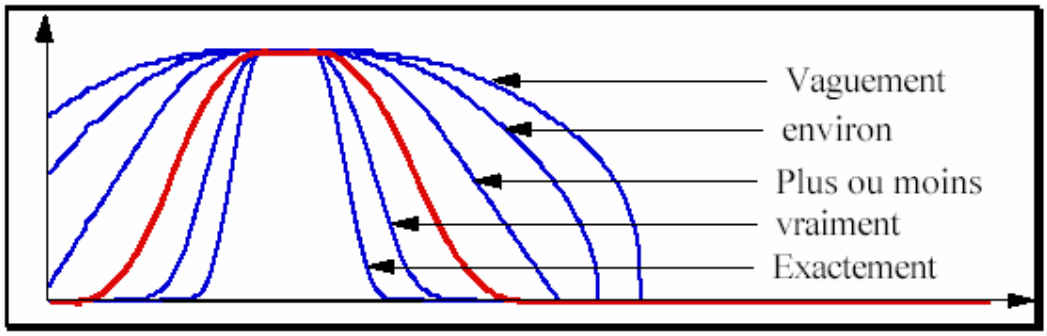


Figure 4. Application d'un opérateur flou sur une propriété (figure issue de [DP97]).

Remarque :

On peut toujours composer un opérateur flou avec un modificateur. L'inverse produit au contraire des énoncés n'ayant pas de sens (« très vraiment »), sauf bien entendu dans le cas où l'on fait intervenir l'opérateur ∅.

La composition de deux opérateurs flous est aussi souvent problématique d'un point de vue sémantique. Seul l'opérateur « vraiment » peut être appliqué plusieurs fois (Il joue le même rôle que l'opérateur « très » pour les modificateurs).

2.4 Calcul d'une α-coupe.

2.4.1 Cas d'une propriété simple ou paramétrée sans modificateur.

Pour pouvoir engendrer des scènes qui répondent aux spécifications du concepteur, nous sommes amenés à passer du qualitatif au numérique. Disposant d'une modélisation des

propriétés sous forme de sous-ensembles flous, nous allons être amenés à calculer des α-coupes pour obtenir des intervalles.

On définit une α-coupe par la formule suivante :

$$A_\alpha = \{x \in E, \mu_A(x) \geq \alpha\}$$

Afin de simplifier l'exploration de l'α-coupe pour la génération des scènes solutions, nous obligeons celle-ci à être un intervalle. On obtient ceci en imposant que le sous-ensemble flou soit convexe [GAC97]. C'est le cas pour une fonction d'appartenance de type L-R. Lorsque l'on effectue une α-coupe A_v pour une valeur $v \in]0,1]$ sur une fonction d'appartenance de type L-R, on a :

$$A_v = [a-\alpha L^{-1}(v) ; b+\beta R^{-1}(v)]$$

Par ailleurs, il est primordial pour nous que l'α-coupe soit facilement calculable. Afin d'avoir une méthode efficace de calcul des α-coupes, dans [JPP02b], nous avons choisi comme fonctions d'appartenance représentant les propriétés des fonctions trapézoïdales.

Les fonctions trapézoïdales correspondent à des sous-ensembles flous convexes et donnent des formules simples pour le calcul effectif des α-coupes.

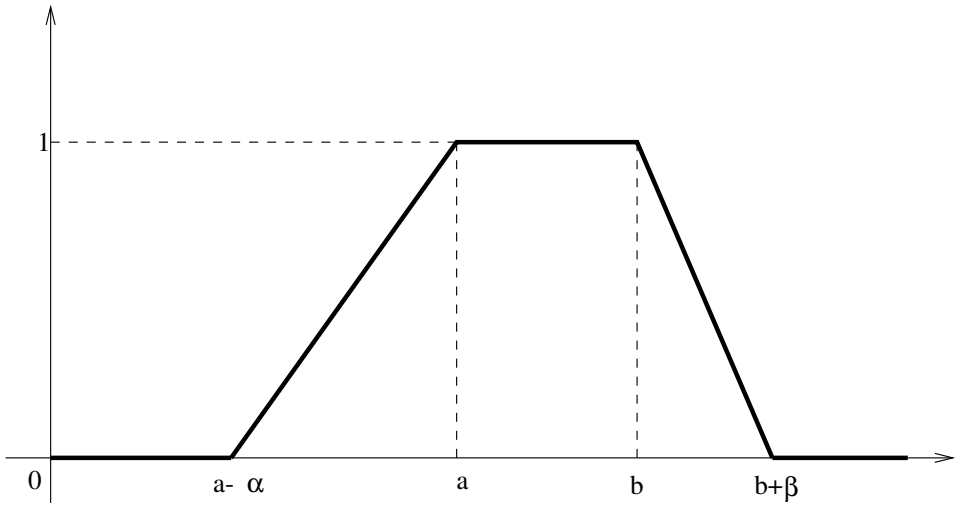


Figure 5. Sous-ensemble flou trapézoïdal.

Les fonctions trapézoïdales sont définies par les fonctions de forme L_1 et R_1 suivantes :

$$\forall x \in \mathfrak{R}, L_1(x) = R_1(x) = \max(0,1-x)$$

Pour les fonctions réciproques des fonctions de forme, comme l'on se restreint à l'intervalle [0,1], on a de façon évidente :

$$L_1^{-1}(x) = R_1^{-1}(x) = 1-x$$

On remarquera que ceci n'est pas la problématique de [DES95] et [DP97] puisque dans ce cas, les sous-ensembles flous ne servent qu'à vérifier si une scène possède ou non une propriété (C'est alors le problème de la calculabilité de la fonction d'appartenance qui se pose).

Dans [DES00], des intervalles numériques sont déterminés à partir des propriétés, intervalles étant des domaines pour des variables dans un problème de satisfaction de contraintes (CSP). L'intervalle calculé correspond au support du sous-ensemble flou modélisant la propriété. Bien qu'il n'y ait aucune justification de ce procédé, on peut penser qu'il a été choisi en raison de sa simplicité. Les bornes du support sont en effet toujours

connues même après l'application d'un ou de plusieurs modificateurs et le problème du calcul d'une fonction inverse ne se pose donc pas avec cette façon de faire.

2.4.2 Cas d'une propriété à laquelle est appliqué un modificateur ou un opérateur flou.

Quand on applique un modificateur à une propriété simple, il n'y a pas de modification des fonctions de forme L et R. Donc la formule de calcul de l'α-coupe A_v donnée au 2.4.1 pour une propriété reste valable.

Par contre, quand on utilise un opérateur flou, les fonctions de forme changent. Pour une fonction d'appartenance f trapézoïdale, on obtiendra :

$$L'_1(x) = R'_1(x) = \max(0,1-x)^j$$
$$= \max(0,(1-x)^j)$$

Une α-coupe A_v pour la nouvelle fonction d'appartenance f' sera donc fournie par la formule :

$$A_v = [a-\alpha' L'^{-1}_1(v) ; b+\beta' R'^{-1}_1(v)]$$

avec :

$$L'^{-1}_1(v) = R'^{-1}_1(v) = 1 - v^{\frac{1}{j}}$$

2.5 La gestion de l'incertitude dans une description.

L'étude précédente a permis de gérer l'imprécision d'une description dans une langue. Nous nous intéressons maintenant à un autre genre de description : des descriptions incertaines.

Elles ont été introduites dans [JPP04] et ont une forme du genre : « Il est plutôt certain que la hauteur du mur est grande ». L'expression « plutôt certain » est un opérateur d'incertitude qui agit sur une propriété.

Nous avons choisi comme ensemble d'opérateurs d'incertitude l'ensemble suivant : {∅, « plutôt », « assez », « peu », « très peu »}.

L'opérateur ∅ signifie que l'utilisateur est certain de sa description (mais, bien entendu, une certaine imprécision demeure, due à la nature imprécise des spécifications déclaratives que nous employons).

Les opérateurs d'incertitude peuvent être employés avec n'importe quel type de propriétés, et en particulier avec les propriétés simples et les propriétés paramétrées que nous avons présentées précédemment.

Pour modéliser une proposition incertaine, nous pouvons réutiliser la notion de sous-ensemble flou. L'action d'un opérateur d'incertitude sur une fonction d'appartenance consiste à contracter celle-ci tout en réduisant son noyau. Cette action est représentée sur la figure 6 :

Un opérateur d'incertitude est caractérisé par un couple de réels $(j,k) \in]0,1] \times \Re^+$. j agit sur la forme de la nouvelle fonction d'appartenance f' et k est un coefficient lié à la réduction de taille du noyau. j = 1 correspond au cas de l'opérateur d'incertitude ∅.

Soit une propriété P complètement déterminée par un quadruplet (α,a,b,β), deux fonctions de forme L et R et un coefficient de translation t.

Soit i un opérateur d'incertitude défini par les coefficients (j,k).

La propriété « i P » a pour fonction d'appartenance f' et pour coefficient de translation t' tels que :

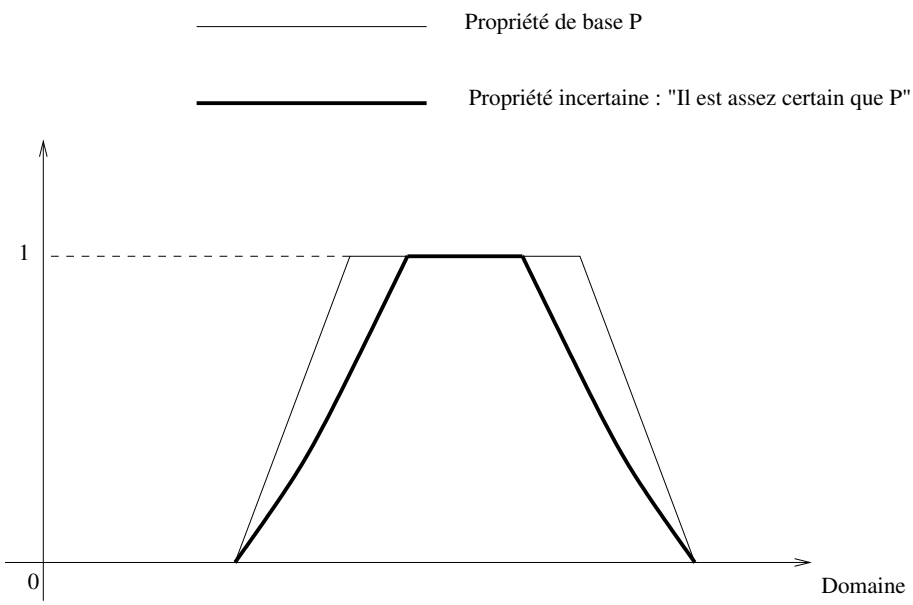


Figure 6. Action d'un opérateur d'incertitude sur une propriété.

$$\alpha' = \begin{cases} \alpha\left(1+\frac{c}{j}\right) si\ j \in\ ]0,1[\\ \alpha \qquad\qquad si\ j = 1 \end{cases}$$

$$\beta' = \begin{cases} \beta\left(1+\frac{c}{j}\right) si\ j \in\ ]0,1[\\ \beta \qquad\qquad si\ j = 1 \end{cases}$$

$$a' = a + \frac{kc\,(b-a)}{2}$$

$$b' = b - \frac{kc\,(b-a)}{2}$$

$$t' = t$$

$$L' = L^{j}$$

$$R' = R^{j}$$

Dans les applications, nous avons choisi comme auparavant c = 10%.

Comme les fonctions de forme changent quand on utilise un opérateur d'incertitude, l'α-coupe A_v associée à la nouvelle fonction d'appartenance f' sera :

$$A_v = [a-\alpha' L'^{-1}_1(v)\ ;\ b+\beta' R'^{-1}_1(v)]$$

avec :

$$L'^{-1}_1(v) = R'^{-1}_1(v) = 1 - v^{\frac{1}{j}}$$

2.6 Exemple.

Examinons une application numérique tirée de [JPP02b].

On considère la scène de la figure 7 :

Figure 7. Une scène d'intérieur.

Le concepteur fournit comme spécification d'ambiance : « Le haut de la table est très éclairé ».

On suppose que l'on dispose d'un moyen de reconnaître les éléments géométriques correspondant au haut de la table. Pour ceci, on peut imaginer qu'il existe un fichier associant des éléments sémantiques de la scène avec des éléments géométriques de celle-ci. Nous ne nous intéresserons pas à ce problème et nous concentrerons uniquement sur la production d'une certaine ambiance (qui passe ici par l'éclairage).

Une propriété simple entre en jeu : « éclairé ». Comme paramètres pour le sous-ensemble flou associé à celle-ci nous pouvons prendre :

$$a = 100\ Watts$$

$$b = 150\ Watts$$

$$\alpha = \beta = 10$$

Nous en inférons la fonction d'appartenance associée :

$$f(x) = \begin{cases} 0 & si\ x < 90 \\ \dfrac{x}{10} - 9 & si\ 90 \le x < 100 \\ 1 & si\ 100 \le x \le 150 \\ 16 - \dfrac{x}{10} & si\ 150 \le x \le 160 \\ 0 & si\ x > 160 \end{cases}$$

Comme on a une propriété positive, on choisit un coefficient t de translation positif :

$$t = 20$$

La propriété souhaité par l'utilisateur est : « très éclairé ». Le coefficient du modificateur est donc : k = 2.

En recourant aux formules de transformations données précédemment, on aboutit ainsi aux valeurs suivantes pour le nouveau sous-ensemble flou :

$$a' = 145\ Watts$$

$$b' = 185\ Watts$$

$$\alpha' = \beta' = 8$$

La fonction d'appartenance f' liée à la propriété modifiée est donc :

$$f'(x) = \begin{cases} 0 & si\ x < 137 \\ 1 - \dfrac{145 - x}{8} & si\ 137 \le x < 145 \\ 1 & si\ 145 \le x \le 185 \\ 1 - \dfrac{x - 185}{8} & si\ 185 \le x \le 193 \\ 0 & si\ x > 193 \end{cases}$$

Les courbes représentatives des fonctions f et f' apparaissent sur la figure 8.

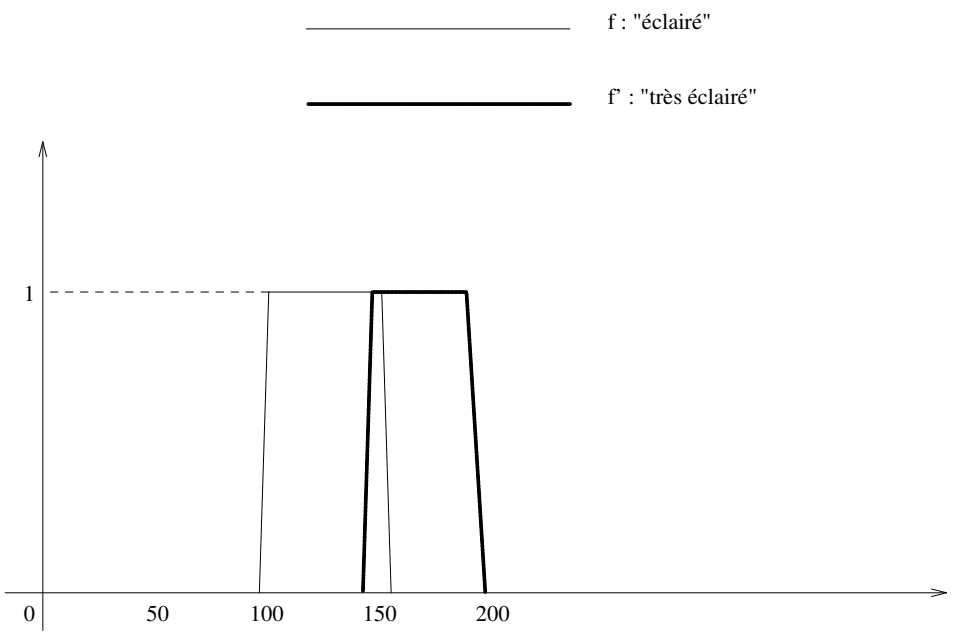


Figure 8. Courbes représentatives des propriétés « éclairé » et « très éclairé ».

En prenant pour seuil d'acceptation $\alpha = 0.5$ pour l'α-coupe, on obtient comme intervalle des valeurs possibles pour l'éclairage du haut de la table :

$$A_{0.5} = [139,189]$$

3 Application à des propriétés d'ambiance dans un paysage en VRML.

Nous allons utiliser la modélisation des propriétés sous forme de sous-ensembles flous pour créer des ambiances brumeuses dans des paysages. De plus, nous allons présenter une méthode de spécification de l'éclairage naturel issue de [SIR96] et [SIR97]. Ces applications ont été présentées dans [JPP04].

Nous avons choisi de travailler avec VRML qui est devenu maintenant un standard du Web3D. VRML possède en effet plusieurs avantages :

- un ensemble de fonctionnalités de haut niveau pour la géométrie de la scène et une prise en compte du rendu par le visualisateur, ce qui permet de ne pas avoir à s'occuper de détails de bas niveau comme avec OpenGL.
- un format de fichier texte, ce qui permet de manipuler facilement les divers mondes virtuels avec des langages de script comme Perl ou Python.
- une spécification publique des possibilités du langage.

Nous pouvons trouver deux types d'outils d'aide à la conception d'un monde VRML :

- des modeleurs géométriques qui s'occupent de l'aspect statique du monde virtuel.
- des modeleurs comportementaux qui permettent de gérer l'aspect dynamique du monde (animation de personnage, prise en compte des actions de l'avatar, etc.) et ceci généralement sans avoir à écrire de code VRML.

Nos travaux se rangent dans la première catégorie en permettant de spécifier de façon déclarative des paramètres liés à l'ambiance d'un paysage en VRML.

3.1 Gestion du brouillard dans un paysage en VRML.

Nous nous proposons d'appliquer le modèle que nous avons développé au sous-chapitre précédent à la gestion du brouillard dans une scène VRML.

Pour gérer le brouillard présent dans un monde, VRML propose le nœud Fog [VRML97], dont les principaux champs sont rappelés dans la spécification suivante :

```
Fog {
        SFColor     color           1 1 1
        SFString    fogTYPE         « LINEAR »
        SFFloat     visibilityRange 0
}
```

Remarques :

- Nous avons supprimé certains détails de la norme VRML pour ne laisser que ce qui touche directement nos travaux.
- La première colonne indique le type du champ, la deuxième son nom et dans la dernière figurent les valeurs par défaut du champ.

Le champ « color » indique la couleur du brouillard. Par défaut, celle-ci est blanche (ce qui correspond à l'effet naturel que nous cherchons à produire).

Le champ « fogTYPE » spécifie la façon dont se mélangent les couleurs (couleur d'un objet de la scène et couleur du brouillard) en fonction de la distance. Nous avons choisi d'affecter la valeur « EXPONENTIAL » à ce champ, ce qui correspond à un brouillard naturel [VRML97].

Le champ « visibilityRange » détermine le seuil de visibilité en mètres (unité de VRML). C'est sur celui-ci que nous allons agir en gérant des spécifications fournies par le concepteur.

Par ailleurs, afin de rester cohérent, le fond de la scène sera mis en blanc.

Les propriétés mises à la disposition du concepteur sont de deux sortes : des propriétés simples et des propriétés paramétrées.

Nous avons introduit la propriété simple : « Le temps est brumeux ». A partir des modificateurs, des opérateurs flous ou des opérateurs d'incertitude, il est possible pour l'utilisateur de changer et d'affiner sa description de l'aspect brumeux de la scène.

Le concepteur dispose aussi une propriété paramétrée prenant la forme : « La limite de visibilité est de 70 mètres ».

Pour chaque description, un intervalle est calculé en utilisant la méthode expliquée précédemment.

Nous avons choisi pour les coefficients intervenant dans la fonction d'appartenance d'une propriété : $\alpha = \beta = 10$

En outre, pour une propriété paramétrée du type : « La limite de visibilité est d mètres », afin de centrer le segment [a,b] sur la valeur d, nous avons choisi l'heuristique suivante :

$$a = d-e$$
$$b = d+e$$

avec :

$$e = \frac{M-m}{10}$$

où : m et M sont les bornes du domaine des fonctions d'appartenance liées au concept d'aspect brumeux de la scène. m et M peuvent être calculés en déterminant la boîte englobante de la scène.

Dans le cas de la propriété simple : « Le temps est brumeux », on détermine les paramètres de la fonction d'appartenance et le coefficient de translation t à partir de la boîte englobante de la scène et des formules de [DES98].

Considérons que le monde VRML repose sur une base rectangulaire de largeur l et de longueur L. Le domaine correspond alors au segment [0,d] (cf. figure 9).

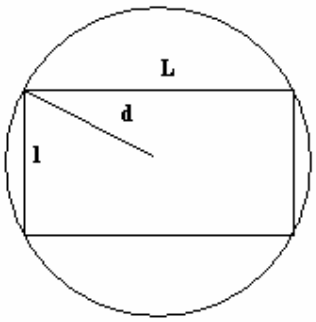


Figure 9. Base rectangulaire de la boîte englobante de la scène et seuil de visibilité maximal.

On a :

$$d = \frac{1}{2}\sqrt{l^2 + L^2}$$

La propriété « Le temps est brumeux » est centrée sur la valeur $\frac{l}{4}$.

D'après les formules de [DES98], on a donc :

$$\begin{cases} \alpha = \dfrac{d}{5} \\ a = \dfrac{5l-d}{20} \\ b = \dfrac{5l+d}{20} \\ \beta = \dfrac{d}{5} \\ t = \min\left(\dfrac{l}{24}, \dfrac{4d-l}{24}\right) \end{cases}$$

Quand nous avons plusieurs descriptions concernant l'aspect brumeux de la scène, l'intersection des intervalles associés à chacune des descriptions est calculée. Si cette intersection est vide, cela signifie que les propriétés demandées ne sont pas compatibles.

Nous allons illustrer avec un exemple la modification que peut introduire un opérateur flou sur une propriété paramétrée. Nous allons comparer les propriétés :

- « La limite de visibilité est de 70 mètres »,
- « La limite de visibilité est exactement de 70 mètres ».

Nous avons pour chaque propriété plusieurs scènes solutions. Elles correspondent à la discrétisation des intervalles obtenus. Chaque scène correspond à une valeur du champ visibilityRange d'un noeud Fog inséré dans le fichier VRML traité.

Sur la figure 10 sont présentées deux scènes correspondant aux deux propriétés précédentes. Nous avons choisi à chaque fois la borne inférieure du segment calculé.

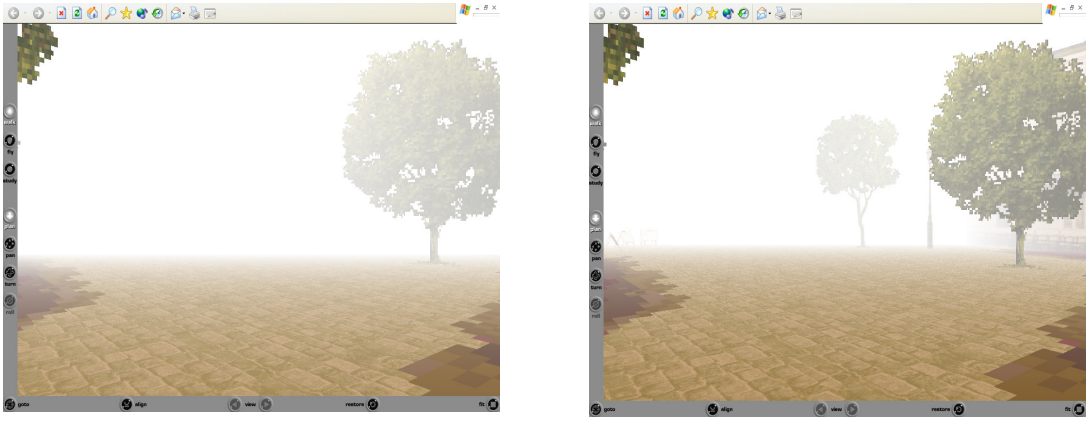


Figure 10. Première image : « La limite de visibilité est de 70 mètres ». Seconde image : « La limite de visibilité est exactement de 70 mètres ».

L'opérateur « exactement » réduit l'imprécision d'une propriété. Dans le contexte du traitement de la nature brumeuse d'un paysage, ceci se traduit donc par un brouillard moins épais.

Pour une propriété donnée, un nombre assez important de solutions sont engendrées (Ce nombre dépendant de la discrétisation choisie). Avec une propriété paramétrée comme « La limite de visibilité est de 70 mètres », on a pour la fonction d'appartenance a = 50 et b = 90.

Sur la figure 11 sont représentées les scènes correspondant à la valeur minimale et à la valeur maximale du coefficient de visibilité pour la propriété « La limite de visibilité est de 70 mètres ».

Nous avons choisi pour la figure 11 les deux scènes correspondant aux valeurs extrêmes des paramètres donnés par nos formules et la différence entre celles-ci est visible. Cependant, on obtient généralement un grand nombre de scènes très proches. Aucun élément de la scène ne permet de les distinguer les unes des autres. Comme il semble difficile de découvrir une heuristique à même d'engendrer des scènes suffisamment différentes entre elles, un outil pour classer les scènes et les présenter de façon rationnelle à l'utilisateur serait donc d'un intérêt certain. Ceci correspondrait à la phase de prise de connaissance en modélisation déclarative.

3.2 Gestion de l'éclairage naturel dans un paysage en VRML.

Nous introduisons maintenant une gestion d'un paramètre influant sur l'ambiance dans un paysage en VRML qui ne fait pas appel à notre méthode de représentation des propriétés par des sous-ensembles flous. Bien entendu, cette approche conserve un aspect déclaratif pour la spécification des désirs de l'utilisateur.

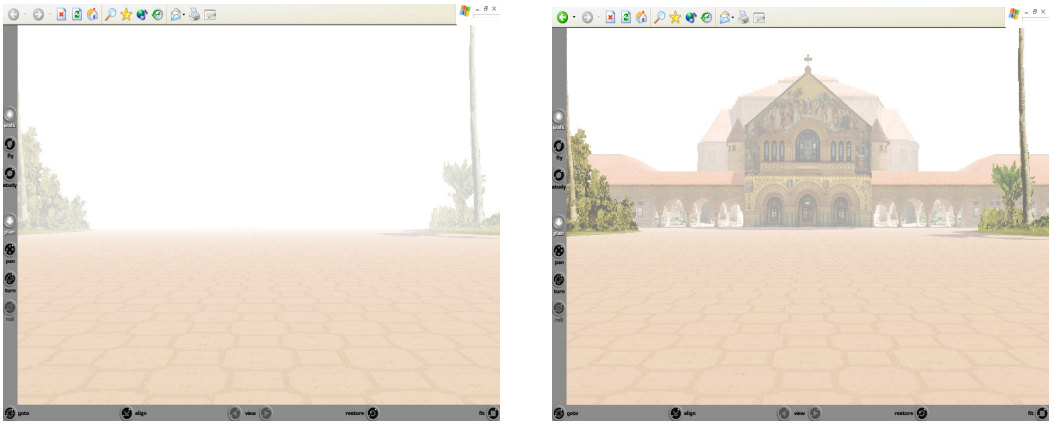


Figure 11. Première image : Première solution à la description « La limite de visibilité est de 70 mètres ». Seconde image : Dernière solution à la description « La limite de visibilité est de 70 mètres ».

Afin de gérer l'éclairage naturel, VRML met à notre disposition un nœud appelé DirectionalLight [VRML97] qui sert à introduire une lumière directionnelle dans la scène. Une spécification simplifiée à nos besoins est :

```
DirectionalLight {
        SFFloat     ambientIntensity    0
        SFColor     color               1 1 1
        SFVec3f     direction           0 0 -1
        SFFloat     intensity           1
        SFBool      on                  TRUE
}
```

Le champ sur lequel nous allons agir est le champ « direction ». Il indique la direction de la lumière. On laissera tous les autres champs à leur valeur par défaut qui est celle indiquée dans la dernière colonne de la spécification ci-dessus.

Afin de conserver une cohérence à la scène, notre méthode modifie les champs skyAngle et skyColor du nœud Background du monde VRML afin de créer un ciel correspondant à l'ensoleillement souhaité par l'utilisateur.

Notre méthode est issue des travaux de [SIR96] et [SIR97] sur l'éclairage naturel d'une scène. L'ensoleillement est décrit grâce au concept de Temps vécu (c'est-à-dire du Temps tel qu'il peut être perçu naïvement et qualitativement), concept introduit dans [SIR96]. Ce concept permet des descriptions de la forme : « La fin de la matinée au printemps ».

[SIR97] propose des formules issues d'études astronomiques pour passer des descriptions du Temps vécu à des intervalles numériques correspondant à la direction du soleil. Cette direction est composée d'un angle azimutal A et de la hauteur H du soleil comme explicité sur la figure 12.

A une description du Temps vécu correspond deux intervalles qui forment une partie de l'hémisphère englobant le monde virtuel. La figure 13 représente une partie T associée à une spécification du Temps vécu. Le point p est le centre du monde VRML dans notre cas.

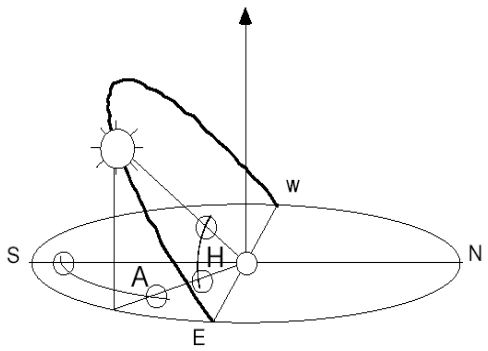


Figure 12. Position du soleil avec le couple (A,H) (figure issue de [SIR97]).

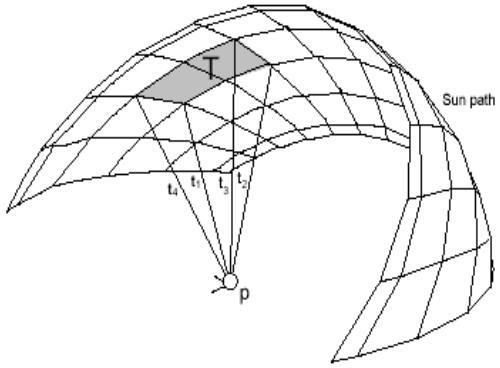


Figure 13. Intervalles décrivant le Temps vécu (figure issue de [SIR96]).

Les possibilités pour produire une description imprécise avec le Temps vécu viennent de compositions d'éléments linguistiques. La figure 14 reproduit les combinaisons possibles.

en début de *en milieu de* *en fin de* *Ø*	X	*la journée* *la matinée* *l'après-midi*	=	*en début de journée* *en fin de matinée* *en milieu d'après-midi* *toute la journée* *etc.*	JOURNEE
			+	*à l'heure h*	
début *mi-* *fin* *Ø*	X	*janvier* *février* *(...)* *décembre*	=	*début janvier* *mi-février* *fin mars* *en décembre*	ANNEE
			+	*les équinoxes* *le solstice d'hiver* *le solstice d'été*	
			+	*l'été* *l'automne* *l'hiver* *le printemps*	

Figure 14. Les possibilités d'expression du Temps vécu (schéma tiré de [SIR97]).

Nous avons appliqué ce formalisme à l'éclairage naturel d'un paysage en VRML. Sur les figures 15 et 16 apparaît la même scène éclairée avec des spécifications différentes. Afin d'engendrer ces scènes, les minima des intervalles calculés ont été sélectionnés.

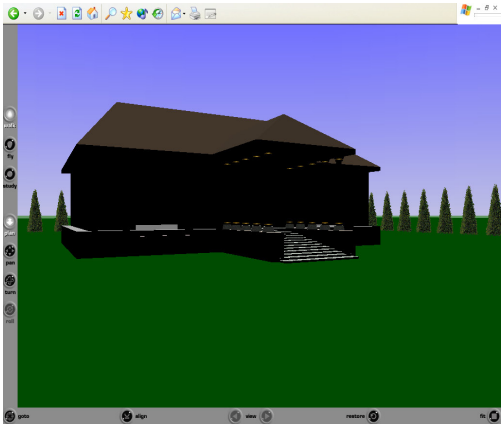


Figure 15. Description de l'éclairage naturel : « En milieu de la journée ».

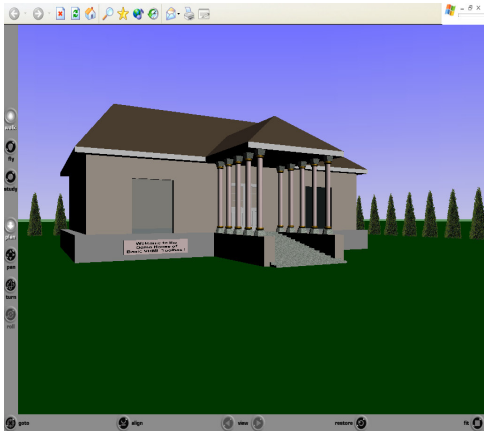


Figure 16. Description de l'éclairage naturel : « En milieu de l'après-midi ».

4 Conclusion sur la création d'ambiance dans des paysages à l'aide d'une spécification déclarative.

Le modèle de gestion des propriétés que nous avons présenté dans ce chapitre peut être réutilisé dans d'autres cadres que les paysages et avec un autre format que VRML. Nous le réemploierons quand nous aborderons le problème de l'ambiance dans une situation plus générale.

Nous pouvons relever des limitations dans les applications que nous avons effectuées de ce modèle à base de sous-ensembles flous à la création d'ambiance dans des paysages en VRML. Ces limitations sont essentiellement dues à des contraintes issues de VRML lui-même. VRML, conçu pour le Web, prend en compte la limitation de la bande passante en proposant un rendu assez sommaire reposant sur les possibilités de base d'OpenGL, sans possibilité d'intervenir sur la phase de rendu pour l'améliorer.

Les problèmes issus de VRML sont :

- L'absence d'ombres qui rend peu réaliste l'éclairage naturel quand la direction solaire change.
- La luminosité incluse dans certaines textures qui n'est pas modifiée quand la direction solaire change et qui persiste alors que l'on peut avoir un brouillard très dense.
- L'impossibilité de prendre en compte des paramètres physiques complexes qui influent sur l'ambiance d'un paysage. Cette impossibilité provient des limitations au niveau du langage et du manque de contrôle du processus de rendu.

Afin de remédier à ces problèmes, plusieurs possibilités sont envisageables :

- Le rendu des visualisateurs VRML repose le plus souvent sur OpenGL. La nouvelle version d'OpenGL (OpenGL 1.5 [SA03]) introduit des fonctions pour gérer les ombres portées. Ainsi, on peut espérer que dans le futur les visualisateurs VRML tiendront compte des ombres dans les scènes.
- L'éclairage contenu dans les textures peut être récupéré et modifié avec la méthode exposée dans [MG97]. Avec une photographie et un modèle 3D de la scène, cette technique permet de retrouver la direction de la source lumineuse (de type directionnelle). On permet ensuite à l'utilisateur de modifier cette direction, puis on produit une image correspondant à cette nouvelle direction. Ceci nous permettrait donc d'obtenir une cohérence entre l'éclairage et les textures du monde VRML. Ce procédé devrait être employé comme une étape de pré-calcul et recourait à un analyseur syntaxique VRML afin de localiser les nœuds Texture dans le source de la scène VRML.
- L'émergence du nouveau standard X3D devrait normalement autoriser une meilleure qualité de rendu. Par ailleurs, un visualisateur comme Contact [CON01] de Blaxxun propose des extensions à VRML qui pourraient permettre de prendre en compte de nouveaux effets au niveau de l'ambiance (Nœud Particles pour gérer un système de particules, nœud MultiTexture pour la cohérence entre l'éclairage et les textures).

Nous avons aussi remarqué que de nombreuses scènes engendrées étaient très similaires. Il semble difficile de trouver un pas de discrétisation tel qu'à chaque scène un nouveau détail apparaisse quand on augmente le seuil de visibilité.

Nous avons cherché à appliquer notre méthode de représentation des propriétés pour créer une ambiance dans un paysage modélisé avec VRML. Nous pourrions l'appliquer à des logiciels spécialisés dans la création de paysages comme Terragen™ [TER04]. Celui-ci permet de gérer beaucoup plus finement l'atmosphère d'un paysage que VRML (qui se veut plus généraliste et adapté au Web). Ce logiciel permet notamment de fixer de nombreux paramètres pour les nuages et l'éclairage naturel. Malheureusement, la manipulation de Terragen se fait uniquement de façon interactive avec des boîtes de dialogues. Le langage de script proposé ne sert qu'à produire des animations. Donc l'exploitation de Terragen ne pourrait se faire que par l'envoi de séries de touches (avec la fonction SendKeys dans un programme VBScript par exemple), technique généralement difficile à mettre en œuvre en raison de problèmes de synchronisation.

Malgré les problèmes mentionnés ci-dessus, la méthode de gestion des propriétés à base de sous-ensembles flous que nous avons développée s'est montrée apte à prendre en compte une spécification déclarative de l'ambiance dans un paysage. De par son aspect généraliste, cette méthode pourra s'appliquer à d'autres scènes que celles représentant un paysage. Elle

permettra à un utilisateur de tester différents types d'ambiance à partir d'une description vague et intuitive.

Partie III : Spécification déclarative de l'ambiance dans le cadre de la radiosité.

1 Introduction.

Nous avons étudié l'ambiance dans le cadre d'un paysage. Nous allons maintenant aborder le problème de la gestion d'une ambiance dans une scène quelconque. La géométrie de la scène est fixée par le concepteur (avec un modeleur quelconque) et nous cherchons à ajouter à la scène des éléments visant à créer une certaine ambiance.

Comme dans le cas d'un paysage, la description de l'ambiance qui sera donnée par le concepteur sera de nature déclarative (c'est-à-dire qu'elle se rapprochera de l'utilisation d'une langue). Cette approche se justifie par le fait qu'une notion comme l'ambiance possède en soi une nature imprécise, floue, pour ne pas dire subjective. Se pose donc la question du vocabulaire que l'on met à disposition de l'utilisateur pour spécifier une ambiance.

1.1 Vocabulaire pour la description d'une ambiance et discussion autour du concept d'ambiance.

Pour le lexique des termes que pourrait manipuler le concepteur afin de décrire une ambiance, nous proposons la taxinomie suivante constituée de trois niveaux :

- Un niveau proche des données physiques (Exemple de notion associée : «brumeux »), relevant de la sensation.
- Un niveau intermédiaire (Exemple de notion associée : « clarté »), plus proche de la perception que la simple sensation.
- Un niveau sémantiquement complexe (Exemple de notion associée : « gaieté »).

Il va de soi qu'au final, pour un traitement informatique, seules demeureront des données physiques. Le problème est alors de savoir s'il est possible de parvenir, par réduction, d'un niveau sémantiquement élevé à un niveau purement physique. Il n'est pas évident qu'une telle opération soit réalisable.

Si chaque niveau est *totalement* déterminé par une configuration d'éléments du niveau inférieur, alors on peut envisager une approche réductionniste. Ainsi, on peut avoir la chaîne de causalités suivante :

gaieté → clarté → scène fortement éclairée → données physiques

Cependant, rien n'indique a priori qu'un tel réductionnisme soit envisageable. Le passage d'un niveau à un niveau supérieur peut occasionner des phénomènes d'émergence. Dans un système où se manifeste l'émergence, un niveau de description supérieur ne se ramène pas à une simple combinaison d'éléments du niveau inférieur. Chaque niveau possède des lois propres qui ne peuvent se déduire du niveau inférieur. On peut décrire une situation où apparaissent des processus émergents par le principe suivant : « Le tout est plus que la somme des parties ». De façon plus rigoureuse, on peut donner la définition suivante : « Une propriété est émergente si elle n'est pas réductible aux propriétés non relationnelles de ses parties » [TEL92].

En outre, même si nous disposions de résultats de psychologie cognitive sur cette question, ils ne seraient pas nécessairement exploitables par l'informatique. Ainsi, aucune des études de de Groot sur des joueurs d'échecs de haut niveau n'est utilisée par les meilleurs programmes d'échecs (Sur ce sujet, on pourra consulter [ASBG02]).

Certains travaux, tels [KPC93], prennent en compte des notions comme celles « d'intimité » ou d' « aspect agréable » qui relèvent donc de la troisième catégorie de notre classification d'un vocabulaire possible pour la description d'une ambiance. Ceci laisse donc penser qu'une conception réductionniste est envisageable (au moins si l'on choisit de façon judicieuse le vocabulaire proposé au concepteur). Cependant, les résultats obtenus par [KPC93] sont sujets à discussion. Les réserves que nous émettons proviennent du fait que dans [KPC93], comme dans nos travaux, la géométrie de la scène est donnée a priori et immuable. De ce fait, il préexiste une certaine ambiance, intrinsèque aux objets situés dans la scène. Il est dès lors illusoire de vouloir produire une ambiance qui serait en contradiction avec cette ambiance liée à la géométrie de la scène.

L'ambiance est le résultat d'une interaction entre l'ambiance intrinsèque des éléments géométriques et celle des autres paramètres (éclairage, milieux participants, etc.). La recherche d'une ambiance est ainsi un processus virtuellement sans fin. La modification d'un paramètre comme l'éclairage entraînera un ajustement au niveau des objets de la scène, ceci afin de s'approcher de l'ambiance souhaitée. Cette modification de la géométrie nécessitera à son tour un ajustement des paramètres d'éclairage pour tenter d'instaurer l'ambiance voulue, et ainsi de suite.

Finalement, l'ambiance sera le résultat d'une synthèse et d'un compromis entre l'ambiance introduite par les objets (c'est-à-dire la géométrie) et les autres constituants de la scène.

En conclusion de ces réflexions, on gardera à l'esprit que la géométrie de la scène prédétermine une certaine ambiance. On ne pourra donc introduire que des éléments compatibles avec cette ambiance, pouvant la renforcer ou l'atténuer.

1.2 L'éclairage inverse.

Les paramètres constitutifs d'une ambiance sont extrêmement nombreux et variés. Le fait de choisir un modèle physique impose déjà une restriction quant aux possibilités de gérer certains de ces paramètres. Pour rester dans un cadre de travail raisonnable, nous allons nous concentrer dans cette étude sur l'éclairage et donc aux paramètres liés à celui-ci. En effet, l'éclairage est clairement une composante majeure de l'ambiance d'une scène. C'est donc le problème de *l'éclairage inverse* que nous allons aborder dans la suite de nos travaux de recherche.

Avec les modeleurs employés couramment, quand un utilisateur veut éclairer d'une certaine façon une partie d'une scène, il est obligé de placer lui-même les sources de lumière et de leur assigner des caractéristiques physiques (comme l'intensité lumineuse) de façon plus ou moins intuitive, en espérant que l'éclairage qu'il désire soit produit lors de la phase de rendu. Comme la grande majorité des outils disponibles ne fournissent aucune aide au concepteur pour placer les sources lumineuses et donner des valeurs adéquates aux paramètres physiques de celles-ci, l'utilisateur perd beaucoup de temps avec de multiples essais avant de parvenir à l'éclairage qu'il souhaite obtenir.

Les algorithmes d'éclairage inverse ont pour ambition de retrouver les caractéristiques principales des sources de lumière (position, orientation, paramètres physiques, etc.) à partir d'un ensemble d'objectifs d'éclairage fournis par l'utilisateur.

Dans le chapitre suivant sera présenté un état de l'art sur les problèmes d'éclairage inverse. Une des conclusions qui ressortira de cette étude est que quand l'interface utilisateur

est assez intuitive, les possibilités d'expression des objectifs d'éclairage sont assez limités (Par exemple, on se restreint à un ensemble fini de positions possibles pour les sources lumineuses, ensemble fixé a priori), et, inversement, quand l'utilisateur peut spécifier des formes d'éclairage de façon sophistiquée, l'interface est de très bas niveau (Il faut déterminer des coefficients pour pondérer l'influence de divers facteurs, ou même écrire des scripts où les contraintes souhaitées par le concepteur doivent être retranscrites de manière complexe). Par ailleurs, le recours fréquent dans les algorithmes à des procédés d'optimisation amène à rejeter des configurations auxquelles le concepteur ne pensait pas à l'origine mais qu'il pourrait juger intéressantes car correspondant à l'idée générale qu'il se fait de l'éclairage qu'il désire.

Les critiques que l'on peut donc formuler à partir de l'état de l'art sur l'éclairage inverse sont semblables à celles que la modélisation déclarative ([PLE95], [LD95], [GAI03]) adresse à la modélisation « traditionnelle ». Il nous paraît ainsi pertinent de fournir à l'utilisateur un outil lui permettant de spécifier de façon déclarative l'éclairage qu'il désire pour une scène donnée et lui fournissant de plus un ensemble de solutions satisfaisant la description qualitative de ses souhaits.

Avant de préciser les différents composants qui pourraient intervenir lors de la construction d'un tel système, il convient d'examiner plus attentivement les contextes où l'éclairage inverse peut jouer un rôle important. Le choix d'un (ou de plusieurs) de ceux-ci va influer sur la conception de l'outil que nous allons détailler par la suite, puisque ce choix correspond à la finalité de l'outil.

Nous distinguerons trois situations :

- L'éclairage réaliste d'un endroit (d'une pièce par exemple) tel qu'on peut l'envisager en architecture. Dans ce cas, les dispositifs d'éclairage sont souvent astreints à suivre des règles strictes (positionnement à des endroits bien définis – classiquement : le plafond, alignement des sources lumineuses, etc.) et les sources d'éclairage sont potentiellement visibles (L'utilisateur peut se déplacer dans la pièce et une source de lumière apparaître dans son champ de vision).
- L'éclairage d'un événement, comme par exemple une exposition dans un musée. On cherche alors à mettre en valeur certains éléments avec un éclairage adapté. Les sources de lumières ne sont plus contraintes de respecter des règles aussi rigoureuses que dans la situation précédente (On n'a pas affaire à un cadre de vie). La présence de sources lumineuses dans le champ de vision de l'utilisateur reste possible, bien qu'elle puisse gêner sa contemplation (pour reprendre l'exemple de l'exposition artistique).
- La recherche d'un bon éclairage pour l'obtention d'une image, dans le cadre d'une publicité par exemple. Ici, on peut introduire des lumières dont les positions et les propriétés physiques seront a priori quelconques, puisque le but est de créer une certaine ambiance que doit retraduire l'image produite. Généralement, et contrairement aux deux cas précédents, *les sources lumineuses n'apparaissent pas sur l'image finale*. De plus, la notion de *point de vue* est maintenant fondamentale. Le point de vue est crucial pour la mise en valeur d'un élément.

On voit donc que dans les deux premiers exemples, il s'agit de rajouter des sources de lumières à une scène que l'utilisateur pourra explorer par la suite. Dans le troisième exemple, la détermination des caractéristiques de l'éclairage s'effectue en vue de la création d'une image. Les contraintes et les objectifs sont donc radicalement différents, et ceci a une répercussion sur les algorithmes d'éclairage inverse que l'on développe.

Les travaux que nous avons accomplis se situent dans les deux premières problématiques. Ceci justifie notamment notre usage de la radiosité comme modèle d'illumination. La radiosité permet en effet à un utilisateur de parcourir une scène avec un simple calcul de visibilité.

1.3 Liens avec la modélisation déclarative.

Nous avons évoqué précédemment la modélisation déclarative. Nos recherches s'inspirent de certains principes de ce domaine de recherche.

La modélisation déclarative a pour but de fournir à un infographiste des outils de haut niveau permettant de lui faciliter le prototypage d'une scène. Ces travaux sont partis de la constatation que les modeleurs « traditionnels » n'offraient que des fonctionnalités très spécialisées (souvent complexes à appréhender) et trop proches de considérations géométriques pour l'utilisateur. La modélisation déclarative se place à un niveau conceptuellement plus élevé que les modeleurs classiques. Le modeleur déclaratif permet de spécifier avec des procédés intuitifs les propriétés d'une scène que désire le concepteur. A partir de ces propriétés, le modeleur déclaratif engendre plusieurs squelettes de scènes qui satisfont les propriétés demandées. Ces ébauches de scènes (constituées, par exemple de boîtes englobantes) seront ensuite finalisées dans un modeleur traditionnel, modeleur qui permettra d'introduire ce qui d'un point de vue conceptuel ne constitue que des détails de la scène.

On voit qu'il existe entre un modeleur déclaratif et un modeleur classique le même genre de relation qu'entre UML et un langage comme C++. A partir de divers diagrammes constitués à l'aide d'UML, un outil de génie logiciel peut produire des squelettes de classes que le programmeur complètera pour assurer les fonctionnalités du logiciel. Notons cependant une différence : la description des scènes étant imprécise (parce que s'appuyant fréquemment sur un sous-ensemble d'une langue), plusieurs prototypes de scènes sont générés par le modeleur déclaratif. Ceci permet de présenter au concepteur des esquisses de scènes solutions auxquelles il ne pensait pas quand il a formulé ses exigences et qu'il peut néanmoins juger intéressantes. La différence existante résulte du fait qu'UML est un ensemble de notations servant à la spécification de logiciels, alors qu'un modeleur convient pour des productions artistiques où des notions difficilement quantifiables interviennent fréquemment.

On distingue habituellement deux types de modeleurs déclaratifs.

Les modeleurs déclaratifs généralistes prennent en charge des scènes quelconques. Par exemple, le modeleur déclaratif MultiFormes [PLE95] permet de spécifier une scène à l'aide de la notion de boîte englobante comme unité spatiale de base, de la notion de positionnement entre boîtes englobantes, et du concept de hiérarchie entre les différents éléments de la scène.

Les modeleurs déclaratifs spécialisés se concentrent eux sur certains types de scènes ou certains aspects d'une scène. Les travaux qui sont présentés ici relèveraient plutôt de cette catégorie.

1.4 Spécification de l'éclairage sous forme déclarative.

1.4.1 Introduction.

Nous allons rechercher dans ce chapitre les différentes propriétés d'éclairage que l'on peut spécifier de façon déclarative. Ces propriétés appartiennent aux familles que nous avons présentées lors du chapitre 1 de la deuxième partie de ce mémoire. Les propriétés seront donc bien évidemment modélisées par des sous-ensembles flous.

Nous reprenons ici les principales idées dégagées dans [JPP02b].

Pour le travail de recensement des diverses propriétés que le concepteur peut avoir envie de manipuler, nous séparerons celles qui se rapportent aux objets à éclairer de celles qui ont trait aux sources de lumière.

On remarquera que nous avons décidé de ne pas essayer de traiter des propriétés complexes liées à des impressions subjectives (comme la notion de « gaieté » par exemple). Nous adoptons une attitude de réserve quant à ce genre de propriétés, suivant les réflexions que nous avons émises au chapitre 1.1.

1.4.2 L'éclairage des objets.

Par le terme « objet », nous entendons un ensemble de carreaux de la scène possédant une unité sémantique explicitée dans un fichier où un nom est associé à cet ensemble de carreaux.

Le concept qui intervient naturellement dans cette partie est celui d'éclairage.

A partir de ce concept, on peut dégager trois propriétés de base : faiblement, moyennement et fortement (Exemple : « A est *faiblement* éclairé »). Le fait de demander « A est éclairé » peut s'interpréter comme « A est *moyennement* éclairé ».

On peut appliquer à ces propriétés de base :

- des modificateurs. Exemple : « A est *assez peu* fortement éclairé ».
- des opérateurs flous. Exemple : « A est *plus ou moins* fortement éclairé ».

Sous certaines conditions, on peut appliquer plusieurs modificateurs et opérateurs flous à une propriété simple. L'utilisateur peut ainsi vouloir spécifier : « A est *vraiment très* fortement éclairé ».

Pour ce qui concerne les propriétés paramétrées, nous ne pensons pas qu'elles soient pertinentes à ce niveau de conceptualisation. Une description telle que : « A a un éclairage *entre 100 et 200* Watts » relève plutôt de la représentation interne sous forme d'intervalles flous des propriétés modifiables.

Il est enfin possible de faire intervenir une négation d'une propriété simple. On peut avoir une spécification telle que : « A *n*'est *pas* très fortement éclairé ». Dans [DP97] est développée une réflexion sur l'interprétation linguistique que l'on doit effectuer de la négation d'une propriété. Celle-ci ne peut être assimilée à la simple négation logique de la propriété. L'utilisateur a plutôt en tête une propriété peu similaire (La notion de similarité de deux sous-ensembles flous est définie rigoureusement dans [DP97]). Un choix est alors proposé à l'utilisateur, choix où on lui présente en premier les interprétations les plus plausibles de la négation qu'il a demandée.

Remarque sur l'interprétation linguistique des propriétés demandées par l'utilisateur :

Considérons une scène dans laquelle deux objets A et B sont très proches. Examinons maintenant les deux spécifications suivantes :

- « A est très fortement éclairé » (1)
- « A est très fortement éclairé et B est très faiblement éclairé » (2)

On peut se demander si en requérant (1), l'utilisateur ne s'attend pas implicitement à ce que ce soit (2) qui soit réalisée. La question qui se pose donc est celle de l'interprétation linguistique du souhait du concepteur. Il est peut-être envisageable de lever l'ambiguïté qui se présente en proposant au concepteur des propositions similaires à (1), d'une façon semblable à celle que [DP97] emploient afin de traiter la négation d'une propriété. Bien entendu, c'est une stratégie qui sera intéressante dans la mesure où A et B seront suffisamment proches (Il

faudra donc avoir un critère pour décider quand considérer qu'une propriété comme (1) doit être précisée).

1.4.3 Les propriétés portant sur les sources de lumière.

1.4.3.1 Introduction.

Pour l'éclairage des objets, nous n'avons dégagé qu'un seul concept. Au contraire, pour les sources lumineuses, il apparaît rapidement que plusieurs notions peuvent intervenir, notions de nature géométrique ou quantitative. La forme de la source lumineuse va avoir une fonction de sélection initiale parmi les concepts qui préciseront ensuite la nature de la source de lumière. De plus, la traduction des propriétés associées à ces concepts en termes d'intervalles flous sera parfois moins pertinente que pour le concept d'éclairage d'un objet.

1.4.3.2 Le concept de forme.

La forme que l'on souhaite attribuer à une source d'éclairage est une caractéristique fondamentale. Si l'on met à disposition de l'utilisateur des spots, certains concepts en relation avec des sources de lumière surfaciques ne s'appliqueront nullement aux spots, et vice-versa (La taille par exemple, ou l'orientation qui a une toute autre signification dans le cas d'un spot). La forme a par ailleurs un impact considérable sur les propriétés liées à d'autres concepts (comme ceux de position, taille et d'orientation). Ainsi, plus la forme est grossière, plus on a de chances de satisfaire une propriété. Un quadrilatère a par exemple moins de contraintes géométriques qu'un carré, et il fournira plus de solutions à la propriété simple : « La source lumineuse est *extrêmement grande* ».

A ce concept de forme, on peut associer un domaine de description D fini. D contiendra des sources lumineuses surfaciques et un spot. On pourrait dans un premier temps se limiter à D = {carré, rectangle, parallélogramme, quadrilatère, disque}$\cup${spot}. On peut considérer qu'une propriété correspond à chaque valeur du domaine D pour le concept de forme. Ces propriétés peuvent être traitées comme des propriétés paramétrées, avec une notion de léger flou pour la forme d'une source de lumière surfacique.

1.4.3.3 Concepts et propriétés spécifiques aux sources d'éclairage surfaciques.

Nous avons déjà évoqué certaines d'entre elles dans le paragraphe précédent pour illustrer la différence entre sources de lumières surfaciques et spots.

La taille d'une source de lumière est un concept auquel on peut rattacher trois propriétés de base : petite, moyenne et grande. On peut appliquer à ces propriétés des modificateurs et des opérateurs flous (les mêmes que ceux que l'on a examiné au 1.4.2 pour les propriétés liées à l'éclairage d'un objet de la scène).

L'orientation d'une source lumineuse peut être un concept intéressant pour le concepteur. Elle doit s'effectuer par référence à une direction dans le plan affine où se situe la source d'éclairage (typiquement l'une des directions parallèles aux côtés d'un polygone modélisant le plafond d'une pièce). On peut alors introduire une propriété paramétrée (Exemple : « La source A fait un angle de 45° avec la direction D »), avec la possibilité de faire intervenir sur celle-ci des opérateurs flous.

1.4.3.4 Concepts et propriétés spécifiques aux spots.

Il y a deux paramètres fondamentaux qui permettent de décrire un spot. Il s'agit de l'axe du cône lumineux produit par le spot et du paramètre de coupure α qui indique comment décroît l'intensité lumineuse quand on s'éloigne de l'axe du spot.

Ces caractéristiques vont correspondre à deux concepts. Les propriétés associées à ces deux concepts seront des propriétés paramétrées sur lesquelles on pourra appliquer des opérateurs flous.

Pour la spécification de l'axe, on devra préalablement se donner une direction de référence. Typiquement, si les sources de lumières sont astreintes à se situer au plafond d'une pièce, on choisira comme direction l'axe orthogonal au plafond.

1.4.3.5 Concepts et propriétés communes à toutes les formes de sources de lumière.

Le nombre de sources lumineuses est un concept de première importance.

On peut dégager à partir de ce concept trois propriétés de base : faible, moyen et important. On peut aussi faire intervenir des propriétés paramétrées (Exemple : « Le nombre de sources lumineuses est 2 »).

Si l'on peut autoriser l'application de modificateurs et d'opérateurs flous sur les propriétés de base, l'expression d'une propriété paramétrée indique plutôt un souhait précis de l'utilisateur à respecter scrupuleusement.

L'alignement des sources d'éclairage (ou d'une partie des sources d'éclairage) est une propriété que l'on retrouve souvent dans les environnements architecturaux. L'alignement est traité comme une propriété de base sur la scène (ou une partie de la scène). Quand le concepteur demande l'alignement d'un certain nombre de sources lumineuses, il n'y a pas d'ambiguïté dans sa spécification. On peut cependant faire intervenir un petit degré d'imprécision pour obtenir des scènes dans le cas où un alignement strict serait impossible à réaliser.

L'utilisateur peut désirer indiquer une zone où placer la source de lumière. On peut distinguer alors deux types de propriétés :

1. Il faut que le logiciel respecte absolument la propriété formulée (Exemple : « Les sources lumineuses doivent être au plafond »).
2. La propriété demandée correspond à une description vague et imprécise de la position de la source lumineuse. La position devra toutefois vérifier les propriétés du premier type (puisqu'elles ont un caractère impératif). Les propriétés du second type vont donc se référer aux zones indiquées par les propriétés du premier type (si l'utilisateur fournit ce genre de contraintes sur la localisation possible des sources d'éclairage), et préciser à l'intérieur de ces zones la position des sources de lumière.

La combinaison de ces deux sortes de propriétés peut aboutir ainsi à une spécification du genre : « La source A est située au Nord du plafond ». Ce qui est équivalent à la conjonction des deux énoncés suivants :

1. « Les sources lumineuses sont au plafond »
2. « La source A est située au Nord »

Le second type de propriétés est une propriété de base, qui peut subir le traitement d'un modificateur ou un opérateur flou (Exemple : «La source A est située *très* au Nord »).

On peut modéliser ces notions de région de façon grossière sur une carte. La figure suivante correspond à la situation où la zone à l'intérieur de laquelle on doit placer la source d'éclairage est un rectangle :

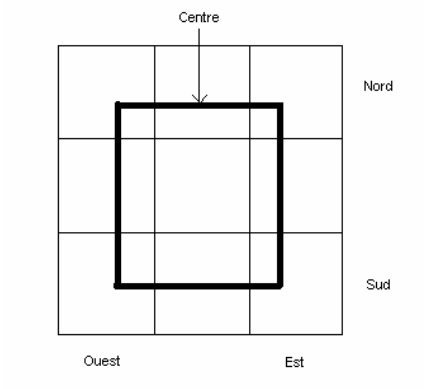


Figure 17. Les différentes régions possibles pour le placement d'une source lumineuse dans une face rectangulaire.

On a ainsi les propriétés de base suivantes : Nord, Sud, Est, Ouest, et Centre. Il est possible de rajouter explicitement des propriétés du style Nord-Ouest plutôt que de construire celles-ci à partir des propriétés plus élémentaires que sont Nord et Ouest.

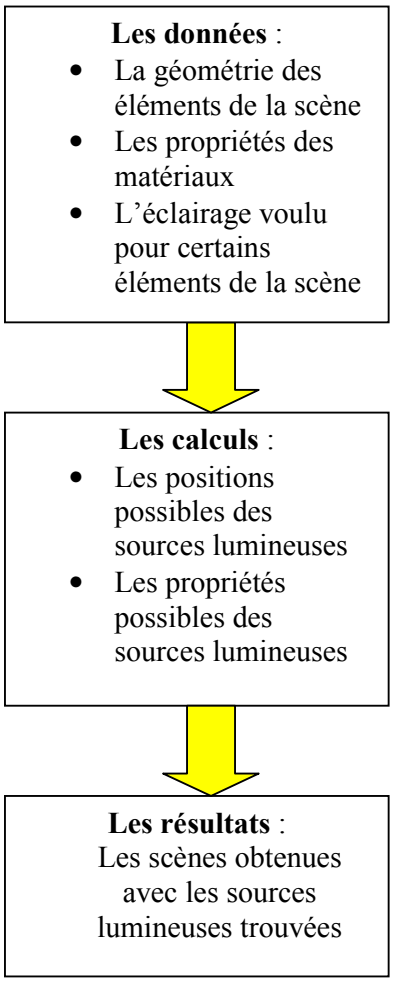
1.5 Vue générale de la suite de cette étude.

Dans les chapitres suivants, nous allons préalablement faire un état de l'art des travaux en éclairage inverse. Nous présenterons ensuite une nouvelle méthode d'éclairage inverse s'appuyant sur une technique de Monte-Carlo. En essayant d'insérer cette méthode dans notre démarche déclarative, nous verrons qu'une utilisation des CSP, employée couramment en modélisation déclarative, ne supprimera pas certains problèmes. A partir de ce constat, nous avons reformulé notre approche, en nous inspirant du paradigme des *Design Gallery* [MAR97] et en utilisant des techniques de visualisation de l'information. La nouvelle méthode que nous proposons se positionnera comme un compromis entre les techniques classiquement employées en éclairage inverse (des méthodes numériques d'optimisation) et le modèle des *Design Gallery*.

2 Etat de l'art sur l'éclairage inverse, avec un aperçu de domaines connexes.

2.1 Introduction.

L'éclairage est un élément fondamental de l'ambiance d'une scène. Pour obtenir un certain type d'éclairage, un utilisateur doit souvent rentrer des paramètres non triviaux dans un logiciel de modélisation et/ou de rendu. Le nombre de ces paramètres et leur complexité fait que le concepteur d'une scène doit souvent entrer dans un long cycle d'essais/modifications pour obtenir l'éclairage qu'il souhaite donner à sa scène. L'éclairage inverse permet à l'utilisateur de spécifier ces désirs en matière d'éclairage (de façon plus ou moins intuitive selon l'interface utilisateur proposée et la méthode d'éclairage inverse mise en œuvre derrière celle-ci). C'est ensuite un algorithme qui va essayer de satisfaire le mieux possible les exigences de l'utilisateur pour l'éclairage qu'il souhaite introduire. La recherche de bonnes valeurs pour parvenir à un certain type d'éclairage n'incombe donc plus à l'utilisateur mais est prise en charge par l'algorithme d'éclairage inverse. On peut ainsi résumer toute la problématique et les principes de l'éclairage inverse par le schéma suivant :



On remarquera qu'avec l'éclairage inverse, on n'obtient généralement pas une seule solution au problème donné. Soit les spécifications demandées sont assez floues pour qu'il puisse y avoir plusieurs façons de les satisfaire, soit les spécifications posent des contraintes trop fortes et on a un ensemble de scènes approchant au mieux les exigences de l'utilisateur.

Puisque dans le cadre de l'éclairage inverse, le concepteur doit indiquer la nature de l'éclairage qu'il désire, il paraît naturel de lui proposer une spécification déclarative de celle-ci. Par ailleurs, étant donné que l'on a comme résultats plusieurs scènes solutions (quand les objectifs ne sont pas contradictoires bien évidemment), nous avons affaire à un domaine qui possède des similarités avec la modélisation déclarative.

Dans les paragraphes suivants, nous allons présenter un état de l'art des méthodes utilisées dans l'éclairage inverse. Préalablement, nous allons examiner des domaines proches de l'éclairage inverse mais dont la problématique diffère de celle de l'éclairage inverse. Nous exposerons aussi quelques techniques employées par des professionnels de l'éclairage afin d'étudier la possibilité (ou non) de les intégrer dans un processus d'éclairage inverse.

2.2 Eclairage inverse et rendu inverse.

Beaucoup de travaux ont été entrepris ces derniers temps dans un domaine connexe à l'éclairage inverse appelé rendu inverse (*inverse rendering*) ou *relighting*. Dans une méthode de rendu inverse, on dispose d'une photographie (qui possède donc un certain éclairage) et du

modèle géométrique correspondant à la photographie[1]. Le but de l'algorithme de rendu inverse est de déduire des informations sur l'éclairage de l'image, puis de produire une nouvelle photographie avec un éclairage différent. Ce qui distingue donc principalement l'éclairage inverse du rendu inverse est que le premier modifie l'espace utilisateur qui ne possède pas a priori d'éclairage, alors que le second travaille à partir d'une photographie afin de changer un éclairage existant.

Nous allons, à titre d'exemple, examiner l'article [MG97] qui propose une méthode de rendu inverse afin de mieux comprendre les techniques qu'elle met en jeu.

A partir d'une photographie, d'un modèle 3D de la scène photographiée et d'une modélisation des caractéristiques de l'appareil ayant produit la photographie, [MG97] présentent une méthode permettant de retrouver les paramètres d'éclairage de la photographie puis de modifier ceux-ci.

[MG97] considèrent la lumière comme étant directionnelle et supposent que la scène est entourée par une sphère englobante. La sphère est divisée en régions qui contribuent diversement à l'éclairage de la photographie. En tenant compte de la linéarité de l'opérateur mathématique lié à l'éclairage, [MG97] calculent les coefficients correspondant à l'influence de chacune des zones de la sphère englobante (grâce à des méthodes d'optimisation comme la méthode des moindres carrés [PTVF92] et la décomposition en valeur singulières d'une matrice [PTVF92]).

Une fois les caractéristiques de la lumière de la photographie découvertes, l'utilisateur peut modifier l'éclairage en changeant la contribution à l'éclairage de chacune des zones de la sphère englobante. A partir des coefficients correspondant à l'éclairage de la photographie, une image est engendrée par un processus de rendu, et l'on crée de même une image avec les coefficients du nouvel éclairage. Le ratio entre la luminosité de chaque pixel des deux images produites est alors appliqué à l'image de la photographie initiale. Ceci permet de conserver certains détails de la photographie originale.

Sur la figure 18 est présenté un exemple de résultats issus de [MG97].

Les transformations de la figure 1 ont pour but de coller la photographie (1) à côté d'une autre (qui n'a pas le même éclairage). Pour cela, on doit donc changer l'éclairage de la photographie (1). On récupère les paramètres d'éclairage de la première photographie et ceux de la photographie à laquelle on veut l'associer. On produit une nouvelle photographie (2) à l'aide de ces paramètres. On peut maintenant faire un montage avec la photographie (2) qui sera cohérent au point de vue de l'éclairage : c'est le résultat observable sur l'image (3).

2.3 Les méthodes d'amélioration de l'éclairage.

Avec l'éclairage inverse, l'utilisateur précise quels sont les objectifs d'éclairage de la scène. Certaines approches veulent au contraire augmenter l'automatisation du processus d'éclairage en améliorant celui-ci grâce à l'optimisation de certains critères. L'éclairage obtenu après l'utilisation d'une de ces méthodes sera meilleur (pour ces critères) que l'éclairage de départ. Typiquement, on pourra distinguer des détails que l'on ne voyait pas avec l'éclairage initial. Mais ceci peut aller justement à l'encontre d'une certaine ambiance que l'utilisateur voudrait introduire dans sa scène.

Ce genre de méthodes sera utile quand un grand nombre de scènes seront à éclairer. L'utilisateur ne pourra pas mettre en place un bon éclairage pour une quantité trop élevée de scènes. Il aura alors besoin d'un outil qui sélectionnera de façon automatique les bons paramètres d'éclairage qui mettront en valeur les caractéristiques des différentes scènes.

[1] Certains auteurs définissent de façon plus restrictive *l'inverse rendering* comme l'ensemble des méthodes recherchant à partir d'une image des paramètres ayant permis de produire cette image.

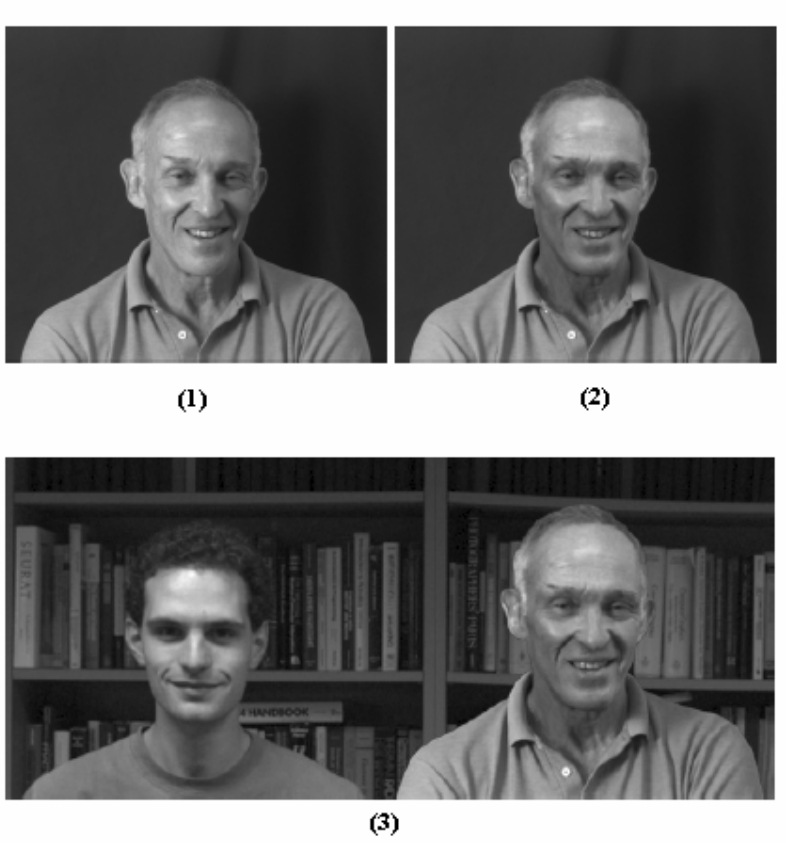


Figure 18. Exemple de résultats obtenus avec la méthode de [MG97] (images tirées de [MG97]).

Bien que nous plaçant dans un contexte différent, nous allons examiner ici deux méthodes améliorant l'éclairage sans intervention de l'utilisateur pour donner une meilleure idée de la nature de ces techniques. Nous allons présenter les travaux de [SL01] et [VS03].

2.3.1 Amélioration automatique de l'éclairage en tenant compte de différentes propriétés de l'image obtenue.

Dans [SL01] est exposée une méthode permettant de modifier l'éclairage pour mettre en valeur l'information contenue dans une scène. A partir de données initiales qui sont la géométrie de la scène, les propriétés physiques des éléments de la scène et un certain point de vue, une ou plusieurs sources lumineuses sont introduites dans la scène afin de faciliter notamment la perception :

- des divers objets (mise en évidence des arêtes par exemple),
- des détails (détails qui pourraient disparaître avec un éclairage trop fort ou trop faible),
- de l'aspect tridimensionnel des éléments de la scène,
- des propriétés des surfaces (rugosité, etc.).

Pour cela, une fonction f_Q est introduite, mesurant la qualité d'une image en calculant la distance entre ses caractéristiques et une valeur idéale pour chacune des propriétés que nous venons d'évoquer. On modifie donc les propriétés de la ou des sources de lumière afin de minimiser cette fonction f_Q. La fonction f_Q a été construite en tenant compte des particularités du système visuel humain (en particulier, la tendance à considérer qu'un objet est éclairé par une lumière monochromatique et placée au dessus de l'objet) et de telle sorte que sa

minimisation amène l'histogramme de la luminance des différents pixels de l'image à posséder de bonnes propriétés statistiques (avec une égalisation de l'histogramme par exemple). Beaucoup d'éléments des différents termes de la fonction f_Q, ainsi que la pondération de ces termes, demeurent toutefois d'origine empirique.

Le modèle physique employé par [SL01] est celui d'OpenGL. Les paramètres qui sont déterminés par le processus d'optimisation sont la position, l'intensité diffuse et l'intensité spéculaire d'une ou plusieurs sources lumineuses ponctuelles.

Trois exemples d'application de la méthode, issus de [SL01] apparaissent sur la figure 19.

Dans les colonnes a et b, des éclairages simplistes ont été choisis. Pour la colonne a, il s'agit d'une source de lumière directionnelle partant du point de vue et dirigée vers l'objet à éclairer. Pour la colonne b, quatre sources lumineuses directionnelles ont été introduites. Leurs origines sont placées sur une sphère englobante de la scène et ces lumières sont dirigées sur l'objet à illuminer. Les coordonnées sphériques (r,θ,φ) des différentes origines des sources lumineuses sont : $\left(R,\frac{\pi}{4},-\frac{\pi}{4}\right)$, $\left(R,\frac{\pi}{4},\frac{\pi}{4}\right)$, $\left(R,\frac{3\pi}{4},-\frac{\pi}{4}\right)$ et $\left(R,\frac{3\pi}{4},\frac{\pi}{4}\right)$ où R est le rayon de la sphère englobante. Les origines sont donc placées respectivement au-dessus et à gauche, au-dessus et à droite, au-dessous et à gauche, au-dessous et à droite de l'objet à éclairer.

Dans les colonnes c et d figurent les résultats obtenus avec la méthode d'amélioration de l'éclairage proposée par [SL01]. Les scènes obtenues diffèrent en raison de configurations différentes. Pour la colonne c, une seule source lumineuse est introduite dans la scène et la fonction f_Q utilisée dans le processus d'optimisation ne cherche pas à placer la lumière au-dessus de l'objet (Le coefficient de pondération de cette composante de la fonction est simplement mis à 0 pour désactiver l'optimisation de cette caractéristique). Pour la colonne d, une seconde source de lumière est ajoutée à la scène et est placée au point de vue. De plus, contrairement à la configuration employée pour les résultats de la colonne c, la phase d'optimisation essaie à présent de donner l'impression que l'objet est éclairé du dessus.

Les éclairages avec une paramétrisation simpliste des sources lumineuses conduisent à des images où peu de détails apparaissent et parfois jusqu'à l'aspect 3D de la scène n'est plus évident (Cas du cube pour l'éclairage de la colonne a). Ces images paraissent par ailleurs trop sombres. Au contraire, les éclairages produits par la méthode de [SL01] mettent beaucoup mieux en valeur les objets à illuminer.

2.3.2 Amélioration automatique de l'éclairage en utilisant la théorie de l'information.

[VS03] proposent une méthode d'éclairage automatique basée sur la quantité d'information que fournit l'éclairage d'une scène. [GUM02] a aussi utilisé la théorie de l'information pour rechercher le « meilleur » éclairage possible, mais les résultats de [VS03] sont de meilleure qualité grâce à une définition plus juste de l'unité d'information définissant l'entropie d'une scène.

[VS03] considèrent des projections de parties de la scène sur une sphère englobante. Ils choisissent comme unité d'information l'aire relative de chaque région dont la couleur diffère des régions qui l'entourent. Ainsi, la valeur de l'entropie est maximale quand toutes les régions auxquelles est associée une couleur ont la même aire relative. Une région de taille importante associée à une couleur unique conduira à une entropie peu élevée pour la scène.

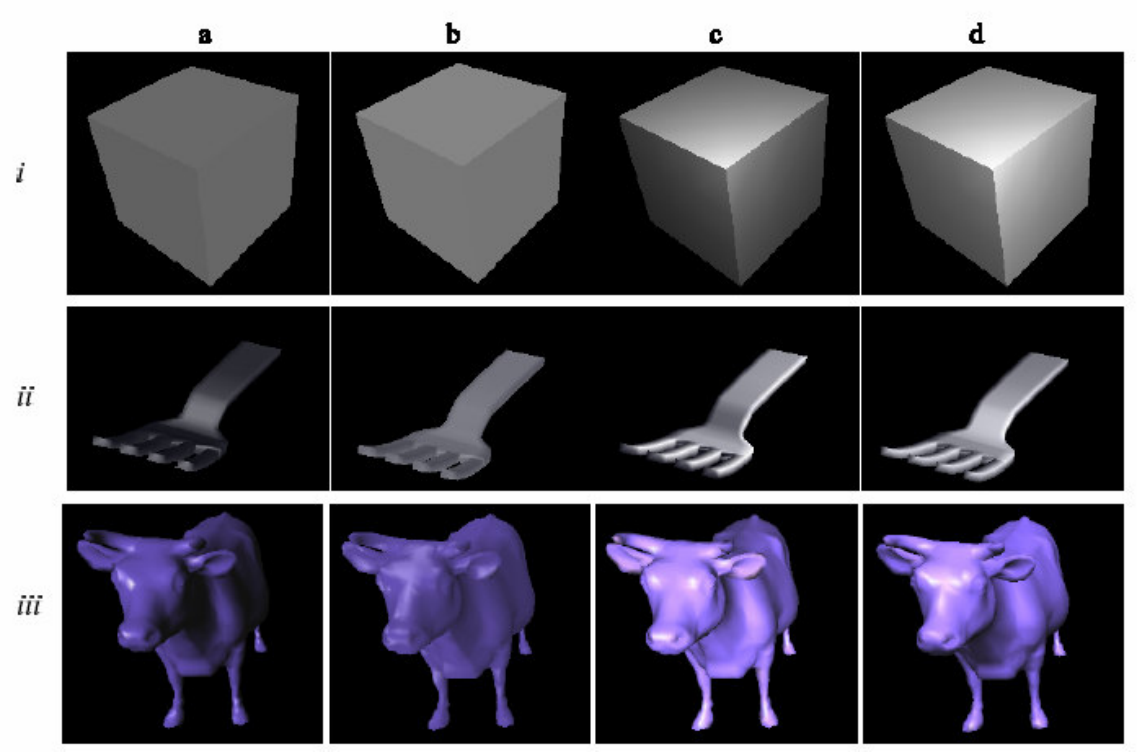


Figure 19. Exemples de résultats obtenus avec la méthode d'amélioration automatique de l'illumination d'une scène (images tirées de [SL01]).

Le procédé chargé de faire correspondre une couleur à une région permet en outre d'éviter de considérer un dégradé sur une grande surface comme une seule couleur associée à cette surface.

Le choix des meilleurs paramètres d'éclairage se fait par un algorithme de recherche exhaustive. Une source lumineuse est placée à chaque endroit de la sphère englobante selon un processus de discrétisation et l'entropie est calculée pour chacune de ces configurations. La position de la source de lumière produisant l'entropie la plus élevée pour la scène sera finalement sélectionnée.

Sur la figure 20 est présentée une comparaison entre des résultats obtenus par [GUM02] et [VS03]. La mesure de l'entropie de [VS03] a été employée pour l'image (a) et celle de [GUM02] pour l'image (b). On remarquera les détails apparaissant sur l'arrière du véhicule dans l'image (a), alors qu'ils sont indécelables avec l'éclairage de l'image (b).

2.4 Techniques d'éclairage professionnelles.

Nous allons examiner dans cette partie quelques principes d'éclairage mis en œuvre par des spécialistes en infographie. Nous nous appuierons sur l'exposé de [KAH96].

Nous présenterons de plus les recommandations de l'IESNA (*Illuminating Engineering Society of North America*) pour les professionnels [IES00].

Un modèle issu des techniques d'éclairage employé au cinéma et en photographie a été transposé au domaine de l'infographie. Comme le remarque [KAH96], ces normes n'ont pas être appliquées de façon systématique. Les professionnels les retrouvent fréquemment mais une conception différente mais cohérente de l'éclairage d'une scène sera préférable à une application irréfléchie des principes présentés dans [KAH96].

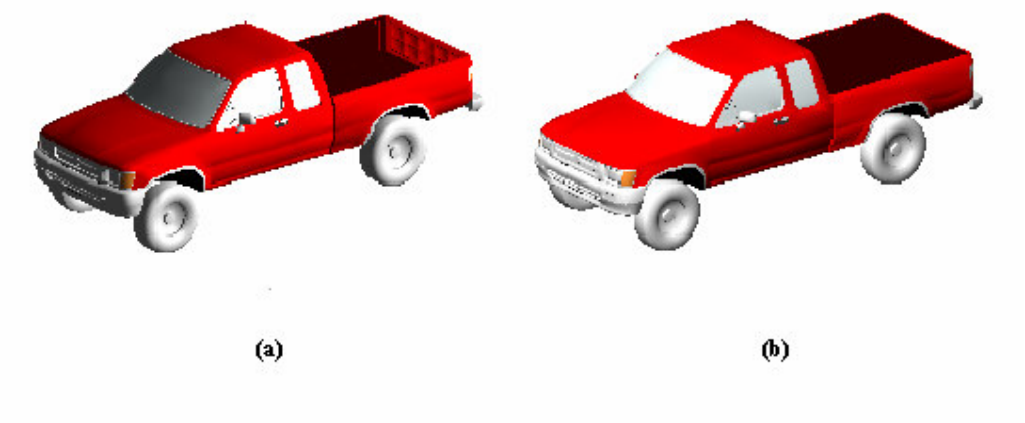


Figure 20. Comparaison entre les résultats de [VS03] et [GUM02].

Trois types de sources de lumières sont utilisés pour mettre en valeur une scène. Chacun de ces types se distingue par sa fonctionnalité plutôt que par ses caractéristiques physiques (caractéristiques qui ne sont pas d'ailleurs déterminées précisément et dépendent des intentions et de l'intuition de l'éclairagiste).

Le premier type de source lumineuse est appelé *key light*. Il assure l'éclairage principal des différents éléments de la scène. Il sert à mettre ceux-ci en valeur. La direction de la source de lumière chargée de cette fonction n'est pas fixée par la norme décrite par [KAH96].

La figure 21 présente l'illumination introduite par une *key light*.

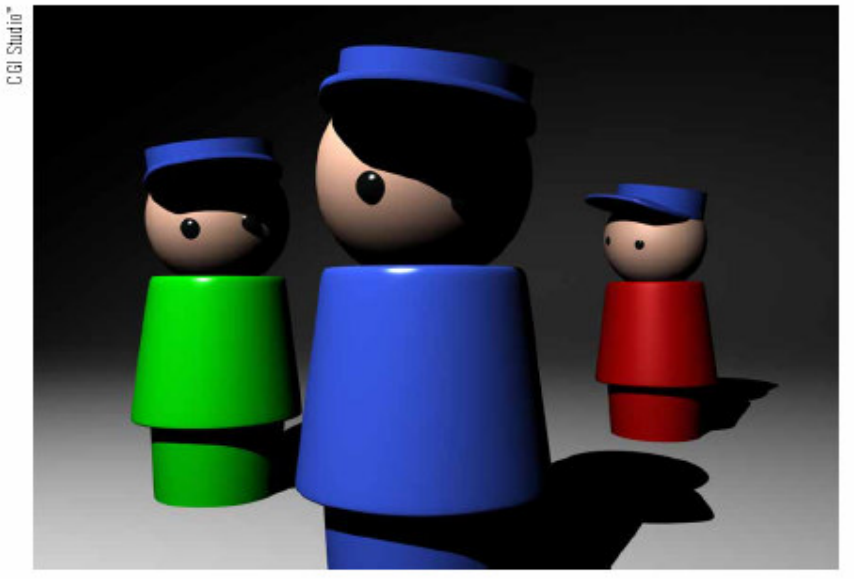


Figure 21. Illumination par une *key light* (Image extraite de [KAH96]).

Le deuxième type de source de lumière est nommé *fill light*. Il sert à adoucir les ombres produites par la *key light*. Il peut s'agir d'une composante ambiante de la scène, d'une source lumineuse supplémentaire ou d'une composition des deux.

Sur la figure 22 est illustrée la fonction d'une *fill light*.

Figure 22. Illumination produite par une *fill light* (Image extraite de [KAH96]).

Le dernier type de source de lumière est appelé *backlight*. Il permet de détacher les objets du fond de la scène.

La figure 23 met en valeur le rôle d'une *backlight*.

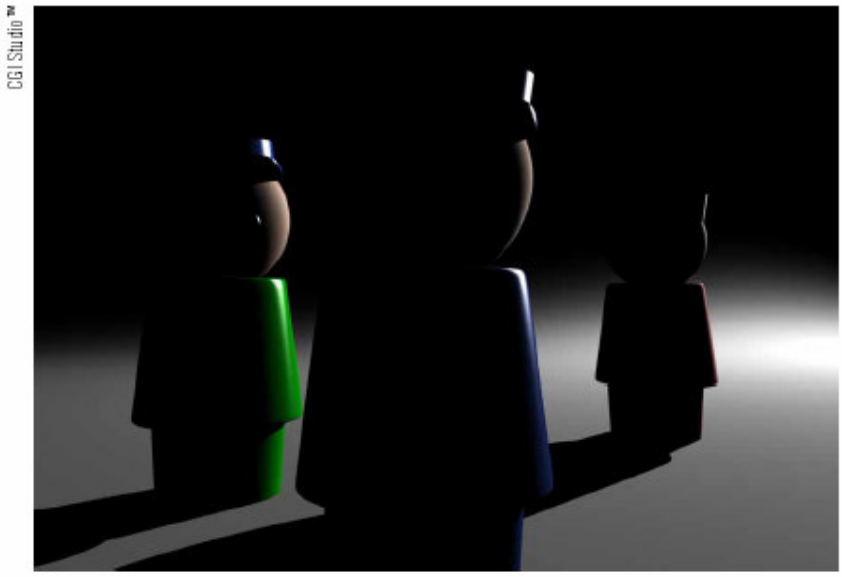


Figure 23. Illumination produite par une *back light* (Image extraite de [KAH96]).

[KAH96] remarque que quand ces principes sont appliqués dans discernement ils aboutissent à un éclairage artificiel. L'origine de ces normes se situe d'ailleurs dans les studios de télévision des années 60 et 70 où l'on a essayé d'obtenir un éclairage fonctionnel, simple et facile à mettre en place.

Le résultat de la combinaison des trois types de sources lumineuses apparaît sur la figure 24.

Figure 24. Combinaison des trois types de sources lumineuses vues précédemment (Image extraite de [KAH96]).

Il est aisé de comprendre que l'exploitation par un algorithme de ces techniques est quasiment impossible tant elles font appel à la sensibilité esthétique du concepteur de l'éclairage de la scène. Il est en outre probable que l'expression sous forme numérique des principes qualitatifs de cette méthode (si elle est possible) aboutisse le plus souvent à la production d'images stéréotypées.

Signalons par ailleurs des articles se focalisant sur des techniques empiriques et qui ont aussi recours à des connaissances d'experts (comme, par exemple [MOE01]). Ce genre d'études trop pragmatiques rend leur exploitation extrêmement délicate. Il est ainsi difficile de comprendre si [MOE01] se contente d'un éclairage direct ou non, et quel est le modèle physique d'éclairement qu'il utilise.

Concluons cette partie en évoquant l'approche fonctionnaliste préconisée par l'IESNA (*Illuminating Engineering Society of North America*) dans [IES00]. En fonction de critères comme le lieu (qui est associé à une tâche à accomplir) et l'âge des personnes occupant la pièce, cet institut publie des valeurs d'éclairage (en lux) qui servent de standards pour les professionnels. On trouve dans cette norme des valeurs pour, par exemples, une chambre d'hôpital, un hall d'hôtel, une cuisine, etc.

2.5 Etat de l'art sur l'éclairage inverse.

2.5.1 Introduction.

2.5.1.1 Définition préalable de quelques termes employés dans l'état de l'art.

Il est d'usage de faire une distinction fondamentale dans les problèmes d'éclairage entre ce que l'on nomme *éclairage global* et *éclairage local* (ou *direct*). L'éclairage global prend en compte tous les phénomènes d'inter-réflexion entre les diverses parties de la scène, ainsi que ceux de réfraction. C'est dans le cadre d'un éclairage global que l'on peut obtenir des effets tels qu'une caustique ou un mélange subtil des couleurs de deux faces adjacentes. L'éclairage local ne tient compte que de l'interaction directe entre une source lumineuse et une surface. Il est le plus souvent dépendant de la position de l'observateur.

Le modèle physique que l'on utilise pour un éclairage global est issu de l'équation de rendu [KAJ86]. Cette équation met en jeu la *radiance*, une grandeur physique dont l'unité est le Watt/m²sr, exprimant l'intensité d'éclairage en un point, en fonction de la direction et de la longueur d'onde de la lumière. La réflexion en un point, depuis une direction et dans une autre, est alors représentée par une fonction appelée BRDF (*Bidirectional Reflectance Distribution Function*).

Si l'on suppose que toutes les surfaces sont parfaitement *diffuses* – ou *lambertiennes* - (c'est-à-dire qu'elles réfléchissent également dans toutes les directions une lumière incidente), que le milieu est non-participant, et en discrétisant la scène en carreaux, on se ramène à un système d'équations plus simple que l'équation de rendu. Cette approche s'appelle la *radiosité*. La radiosité est aussi le nom de la principale grandeur physique qui intervient alors ; elle s'exprime en Watt/m². Les sources lumineuses intervenant en radiosité se caractérisent par l'*émittance*, valeur en Watt/m² spécifiant l'émission spontanée d'énergie lumineuse de ces sources. La fraction d'énergie réfléchie par un carreau est indiquée par la *réflectivité*, une quantité comprise entre 0 et 1.

Pour ce qui concerne l'éclairage direct, le modèle le plus employé est le *modèle de Phong*. Il s'agit d'une technique empirique pour déterminer l'intensité lumineuse en un point de la scène. On décompose l'éclairage et la surface éclairée en 3 composantes :

- La composante *ambiante*, qui sert comme approximation des inter-réflexions entre les différents éléments de la scène.
- La composante *diffuse*, qui donne une apparence de réflexion mate. Elle est indépendante de la position de l'observateur.
- La composante *spéculaire*, qui correspond aux effets de brillance, qui est dépendante de la position de l'observateur, et que l'on peut observer avec des matériaux comme le plastique.

2.5.1.2 Les problèmes d'éclairage inverse.

Nous avons vu qu'en infographie, les problèmes d'éclairage inverse consistent à déterminer à partir d'une scène donnée une ou plusieurs sources lumineuses afin d'obtenir un certain type d'éclairage. Un algorithme fournit les diverses caractéristiques de sources lumineuses qui, intégrées à la scène, permettront de créer une ambiance lumineuse souhaitée par le concepteur. Les caractéristiques dont il est question sont par exemple :

- La position de la source lumineuse.
- Les paramètres physiques de la source d'éclairage, comme l'intensité lumineuse.
- La nature géométrique de la source lumineuse (sa forme).
- Etc.

Le plus souvent, pour un éclairage demandé par l'utilisateur, il n'y pas une seule solution satisfaisante possible. Un ensemble de scènes correspondant plus ou moins aux désirs du concepteur est généralement engendré par un algorithme d'éclairage inverse.

Afin de préciser le problème que l'on étudie, on est amené à spécifier trois caractéristiques fondamentales qui vont influer sur les algorithmes proposés :

1. La nature de l'éclairage (direct ou global).
2. Le modèle physique d'éclairement (modèle de Phong, radiosité, calcul de la radiance, etc.).
3. Les contraintes a priori sur les sources lumineuses (La position des sources lumineuses peut être fixée, et alors seuls les paramètres physiques de ces sources

sont à trouver. On peut aussi avoir un ensemble de positions autorisées, ou tout simplement aucune contrainte sur les positions des sources d'éclairage).

On peut ainsi classer les approches développées pour résoudre les problèmes d'éclairage inverse à l'aide de ces différentes caractéristiques.

De plus, on remarquera qu'il y a une dépendance entre le premier et le deuxième critères retenus pour la taxonomie des problèmes d'éclairage inverse : certains modèles physiques ne permettent pas la prise en compte de l'illumination globale.

2.5.1.3 La nature de l'éclairage.

On peut s'intéresser à un éclairage direct où l'on cherche à déterminer une ou plusieurs sources lumineuses qui illumineront directement une ou plusieurs parties de la scène. On peut aussi tenir compte en supplément de la réflectivité des autres composants de la scène, et l'on s'intéresse alors à un problème d'éclairage global.

Il est clair qu'un problème d'éclairage direct est plus simple à traiter, puisque moins de paramètres entrent en jeu. Il y a toutefois des cas où un algorithme basé sur l'éclairage direct ne peut remplir les objectifs d'éclairage fournis par l'utilisateur (la conjonction de ceux-ci pouvant conduire à une contradiction : par exemple une zone qui se retrouverait éclairée fortement, bien que l'on voudrait qu'elle le soit peu), alors qu'un algorithme recourant à un éclairage indirect le pourrait.

2.5.1.4 Les modèles physiques d'éclairement.

Les modèles ayant été le plus souvent employés jusqu'à présent dans le cadre des recherches sur les problèmes d'éclairage inverse sont :

- Le modèle de Phong (dans le cas d'un éclairage direct).
- La radiosité (qui intervient naturellement dans le cas de l'illumination globale, mais qui peut très bien être le modèle physique sous-jacent à un problème d'éclairage direct inverse).
- Des modèles de calcul de la radiance (pour l'illumination globale).

2.5.1.5 Les contraintes a priori sur les sources lumineuses.

Ces contraintes concernent principalement la position des sources lumineuses dans la scène, mais d'autres peuvent intervenir (par exemple l'orientation d'une source lumineuse).

Les approches des problèmes d'éclairage inverse considèrent :

- Soit que les sources lumineuses sont fixes (et alors seules leurs caractéristiques physiques sont à déterminer).
- Soit qu'il y a un ensemble de sources lumineuses *potentielles* dont les positions sont fixées a priori (Par exemple, dans le cadre de la radiosité, un ensemble de carreaux ayant une émittance éventuellement non nulle).
- Soit qu'il n'y a aucune restriction sur la position des sources lumineuses, hormis le fait que celles-ci ne peuvent sortir d'une boîte englobante (généralement celle de la scène) et qu'elles ne peuvent coïncider avec les autres objets de la scène.

Les contraintes ci-dessus sont souvent imposées au concepteur par l'algorithme employé dans le logiciel d'éclairage inverse. Elles font partie des hypothèses de travail de l'algorithme d'éclairage inverse. Mais le logiciel peut aussi permettre au concepteur de rajouter des contraintes correspondant à ses désirs en matière d'éclairage de la scène. Ainsi, l'utilisateur peut limiter la position des sources lumineuses à rechercher dans une région de la scène, ou imposer une orientation aux sources lumineuses, etc.

2.5.1.6 Remarque touchant la structure de cet état de l'art.

Dans la suite de cet état de l'art, nous prendrons le modèle physique d'éclairement comme critère principal pour classer les diverses études effectuées jusqu'ici dans le cadre des problèmes d'éclairage inverse. Ce choix est surtout lié au fait que nos travaux se situent dans un modèle bien précis : celui de la radiosité.

Une autre compilation a été menée à bien sur les problèmes inverses (et pas simplement les problèmes d'éclairage inverse).

[PP01a], [PP01b] et [PP03] proposent une classification différente de la notre pour les problèmes inverses liés à l'infographie. Partant de l'équation de rendu retranscrite dans le cadre de l'analyse fonctionnelle par [ARV95], ils établissent leur taxinomie en se basant sur les différents opérateurs linéaires et fonctions intervenant dans l'équation de rendu. Suivant les connaissances dont on dispose a priori sur ces opérateurs, ils distinguent trois catégories de problèmes inverses :

1. Les problèmes d'émittance inverse, où les sources lumineuses ont des positions fixées et la géométrie de la scène est intégralement connue.
2. Les problèmes de géométrie inverse, où il s'agit de déterminer la position des sources lumineuses ou celles des objets réfléchissant la lumière. On suppose alors connues les propriétés optiques des éléments de la scène.
3. Les problèmes de réflectométrie inverse, où l'on calcule la réflectance d'éléments de la scène dont les positions sont fournies au départ. Dans ce cas, on dispose de toutes les informations sur les sources lumineuses.

Les articles que nous allons étudier relèvent de ce que [PP01a] et [PP01b] nomment problèmes de géométrie inverse (plus exactement de la partie de ces problèmes concernant les sources d'éclairement), et des problèmes d'émittance inverse.

2.5.2 Articles se plaçant dans le cadre du modèle d'illumination de Phong.

2.5.2.1 Détermination de certains paramètres d'éclairage à partir de zones fortement éclairées et des volumes d'ombre.

a) Utilisation de zones fortement éclairées (*highlights*).

Dans [PF92], les auteurs s'intéressent au terme spéculaire apparaissant dans le modèle de Phong. En exprimant la composante spéculaire sous la forme de Blinn, on a :

$kI\left(\vec{N}.\vec{H}\right)^{\alpha}$ où :

- k est la réflectance spéculaire,
- I est l'intensité lumineuse,
- $\vec{N}$ est le vecteur normal unitaire au point étudié sur une surface,
- $\vec{H}$ correspond à la bissectrice entre le vecteur unitaire $\vec{E}$ indiquant la direction de l'observateur (depuis le point étudié) et le vecteur unitaire $\vec{L}$ indiquant la direction de la source lumineuse (depuis le point étudié),
- α est l'exposant spéculaire (Il correspond au taux de diminution de la réflexion spéculaire depuis le point ayant une intensité lumineuse maximale. Dans la pratique, des valeurs proches de 100 pour α sont assez courantes).

La technique proposée permet de déterminer $\vec{L}$ et α. On détermine donc la direction d'une lumière directionnelle unique.

L'utilisateur sélectionne un point correspondant à l'intensité lumineuse maximale. On en déduit alors la valeur de $\vec{L}$ maximisant le terme spéculaire avec la formule suivante :

$$\vec{L} = 2\left(\vec{H}.\vec{E}\right)\vec{N} - \vec{E}$$

Afin d'affecter une valeur à l'exposant α, l'utilisateur choisit alors un autre point où l'on considère que l'expression $\left(\vec{N}.\vec{H}\right)^{\alpha}$ atteint un seuil τ (τ est une valeur numérique fixée au préalable). On en déduit ainsi la valeur de α :

$$\alpha = \frac{\ln(\tau)}{\ln\left(\vec{N}.\vec{H}\right)}$$

Avec ces caractéristiques, on remplit la région autour du premier point (ayant une intensité maximale) et du second point, avec τ comme borne. Ce processus détaillé dans [PF92] permet d'afficher la zone fortement éclairée en essayant d'éviter d'avoir à calculer les normales de tous les points de la surface éclairée.

On peut ensuite introduire éventuellement d'autres zones fortement éclairées sur la surface considérée, avec la même valeur pour le coefficient α (En effet, α correspond à une propriété physique de la surface et non de la source lumineuse). Pour cela, il suffit de choisir un autre point d'intensité maximale, puis l'algorithme calculera la normale en ce point et remplira ensuite la zone fortement éclairée autour de ce point.

Sur la figure 25 apparaît le contour d'une zone fortement éclairée sur un patch de la théière. Le segment pointé vers la théière indique le point d'intensité maximale et la direction de ce segment correspond à la direction de la source de lumineuse produisant la zone fortement éclairée.

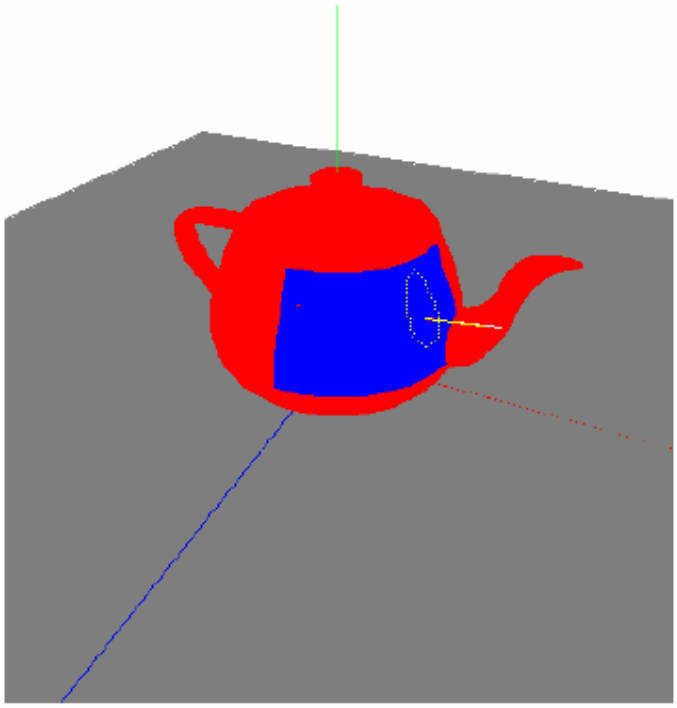


Figure 25. Détermination de la direction d'une source de lumière et de l'exposant spéculaire par la méthode de [PF92] (Image extraite de [PF92]).

[PF92] examinent l'extension de leur méthode à des sources lumineuses ponctuelles, linéaires ou polygonales. En choisissant un plan, à chaque point sélectionné sur la surface à

illuminer va correspondre un point de ce plan, point étant l'intersection d'un vecteur $\vec{L}$ calculé comme précédemment et du plan. On pourrait ainsi construire progressivement une source de lumière de forme polygonale. Mais il faudrait s'assurer que chaque modification de la forme de la source lumineuse n'entraîne pas des contradictions avec le fait que les points précédemment choisis sur l'objet éclairé doivent rester des points où l'intensité lumineuse est maximale. C'est pourquoi [PF92] préconisent plutôt l'utilisation de volumes d'ombres pour créer des sources lumineuses autres que les lumières directionnelles.

b) Utilisation de volumes d'ombre.

En permettant à l'utilisateur de travailler sur des volumes d'ombre, [PF92] parviennent à déterminer des sources lumineuses de divers types (sources directionnelles, ponctuelles, linéaires ou polygonales) à partir des ombres apparaissant dans la scène.

Il est à noter que les volumes d'ombre étant indépendants de la position de l'observateur – contrairement aux effets spéculaires à la surface d'un objet -, ceci facilite la saisie de points dans un espace tridimensionnel (puisque l'on peut gérer simultanément plusieurs fenêtres correspondant à différentes vues).

α) Cas des sources lumineuses directionnelles.

Pour indiquer une direction, il suffit simplement à l'utilisateur de fournir deux points.

Une fois que le volume d'ombre est déterminé par la direction, le concepteur a la possibilité de le déplacer (et donc de modifier la direction de la source de lumière directionnelle). A cette fin, il sélectionne un point P_1 sur la frontière du volume d'ombre. Le point P_2 sur l'objet O créant la zone d'ombre est alors déterminé. En changeant ensuite la position de P_1, on modifie en conséquence la direction d'éclairage. Ce processus est illustré sur la figure 26.

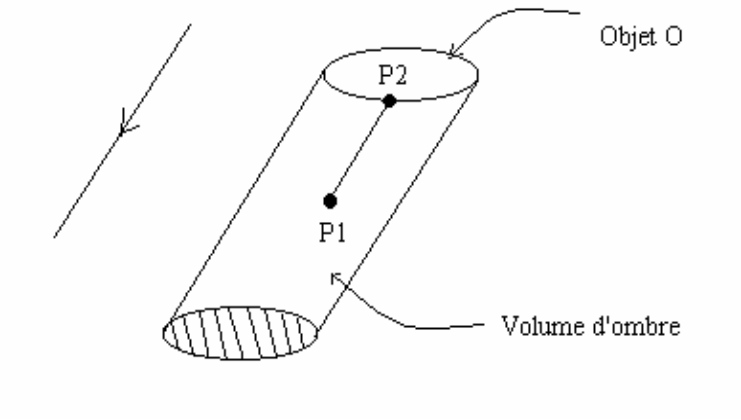


Figure 26. Modification de la direction d'une source de lumière directionnelle à l'aide d'un volume d'ombre.

Une mise en œuvre de ce procédé apparaît sur la figure 27 issue de [PF92]. Le volume d'ombre est affiché en procédant seulement à la projection des vertices formant les bases du cylindre constituant la scène.

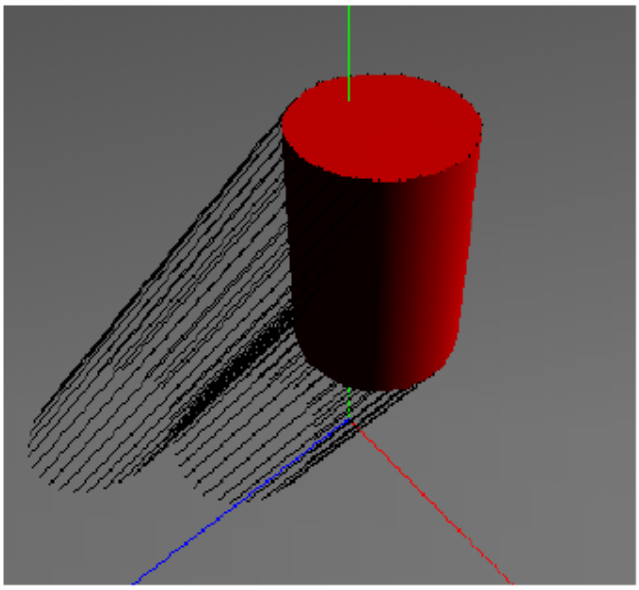


Figure 27. Création d'une lumière directionnelle à l'aide du volume d'ombre produit par celle-ci (image tirée de [PF92]).

β) Cas des sources lumineuses ponctuelles.

Le procédé permettant d'introduire une source lumineuse ponctuelle est une extension de celui servant à la création d'une source lumineuse directionnelle.

L'utilisateur sélectionne un point sn_1 sur la silhouette du volume d'ombre. Le point sn_2 situé sur l'objet O et tel que la droite (sn_1 sn_2) soit parallèle à la direction d'éclairage actuelle est ainsi déterminé. Ces deux points, sn_1 et sn_2 , sont considérés comme fixes pour la suite du processus de calcul d'une source de lumière ponctuelle.

L'utilisateur choisit ensuite un autre point s_1 sur le bord du volume d'ombre. A ce point s_1 correspond un point s_2 placé sur l'objet O (de la même façon qu'à sn_1 correspond sn_2).

En considérant s_2 fixe et en déplaçant s_1 en s'_1, on fixe alors en P la position d'une source lumineuse ponctuelle.

Ce procédé est illustré sur la figure 28.

On peut ensuite modifier la source de lumière ponctuelle à l'aide du volume d'ombre que l'objet O engendre, ceci d'une façon similaire à celle vue au a.

γ) Cas des sources lumineuses linéaires ou polygonales.

Pour ce genre de sources lumineuses, on traite chaque sommet de la source d'éclairage comme une source de lumière ponctuelle. On considère ainsi qu'il y a un volume d'ombre associé à chaque sommet de la source d'éclairage.

La détermination du volume d'ombre produit par la source lumineuse est assez complexe puisqu'elle fait appel :

- à une décomposition de la source lumineuse et de l'objet O en parties convexes (s'ils ne le sont pas au préalable).
- à des calculs d'enveloppes convexes.

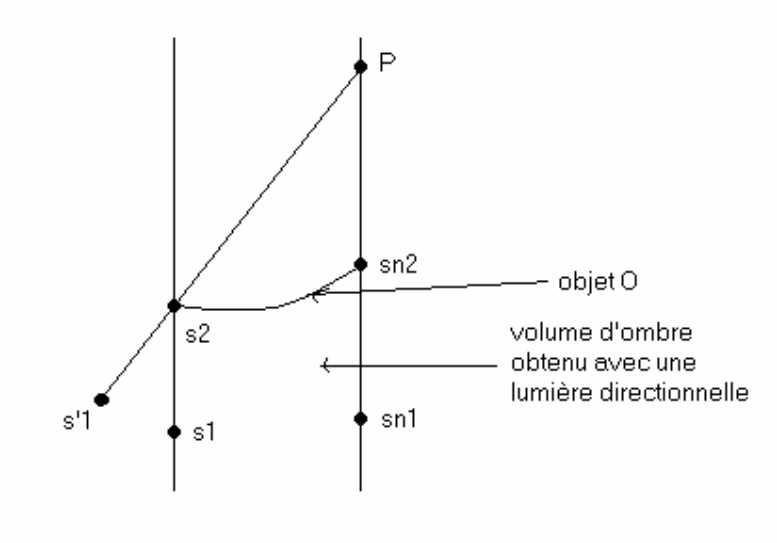


Figure 28. Création d'une source de lumière ponctuelle.

Sur la figure 29 est présenté le résultat de l'éclairage d'un cône par une source lumineuse triangulaire. Les traits rouges correspondent à l'ombre (qui est située à l'intersection des volumes d'ombre produits par chaque vertex de la source de lumière). Les traits noirs correspondent à la silhouette de la pénombre (qui est la différence entre l'union des enveloppes convexes des volumes d'ombre engendrés par les sommets de la source de lumière et la zone d'ombre).

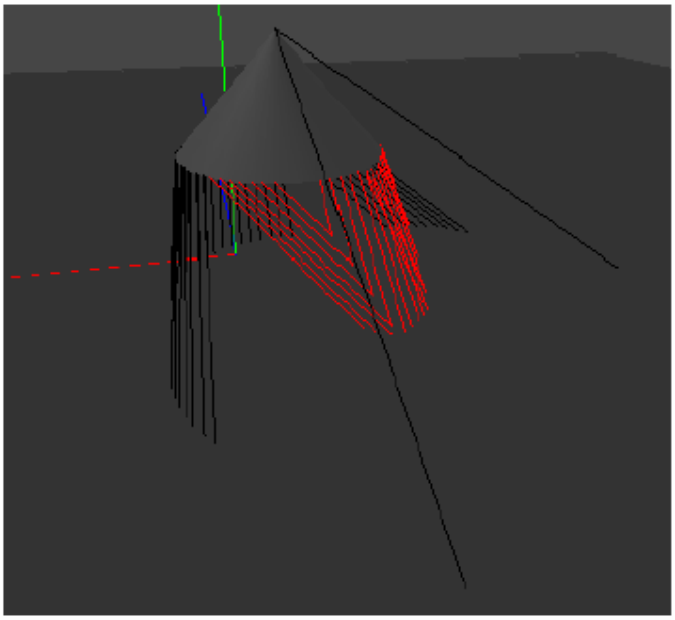


Figure 29. Ombre et pénombre crées par l'illumination d'un cône par une source de lumière triangulaire (image tirée de [PF92]).

c) Remarques.

L'algorithme calculant une source lumineuse grâce à un volume d'ombre ne dépend pas des spécificités du modèle de Phong. Cependant, il n'est valable que pour un éclairage direct des volumes situés dans la scène.

D'autre part, l'utilisateur ne peut aucunement manipuler directement les frontières des ombres portées des objets. Il est contraint de travailler sur des volumes d'ombre, ce qui constitue une méthode de conception moins simple et intuitive. L'approche qui va être développée maintenant, se basant sur des esquisses d'ombres portées répond à cette attente.

2.5.2.2 Détermination de sources de lumière à l'aide d'esquisses d'ombres portées et d'esquisses de zones fortement éclairées.

a) Utilisation d'esquisses.

[PRJ97] prenant en compte les réserves exprimées dans les remarques précédentes ont développé un logiciel permettant d'obtenir des sources lumineuses de diverses natures à partir d'esquisses (de lignes polygonales) marquant les contours de l'ombre portée d'un objet, ou d'esquisses de zones fortement éclairées.

b) Lumières obtenues à l'aide d'esquisses de zones d'ombre.

α) Modélisation du problème.

Dans l'approche retenue, l'utilisateur trace des lignes polygonales (des « traits ») dont on appellera les points de contrôle des *points d'esquisse*. On considère d'autre part un objet de la scène - appelé *bloqueur* - qui servira à produire une ombre correspondant aux esquisses. L'algorithme doit garantir que chaque esquisse se situe à l'intérieur de la zone d'ombre effectivement produite (Avec le processus d'optimisation employé, les esquisses se retrouvent en pratique sur les bords de la zone d'ombre).

Dans un premier temps, pour l'exposé de la méthode, on supposera que la scène se limite à un bloqueur convexe.

β) Cas d'une source lumineuse ponctuelle.

L'algorithme proposé trouve une position pour une source de lumière ponctuelle afin que l'éclairage direct du bloqueur crée l'ombre dont l'utilisateur a donné une esquisse. Il s'agit d'un algorithme employant une technique d'optimisation.

A chaque point d'esquisse on associe un cône, cône dont le sommet est le point d'esquisse et la base la silhouette du bloqueur observé depuis le point d'esquisse. La partie du cône située au-dessus du bloqueur forme la région des positions valides pour une source lumineuse. La figure 30 illustre ce procédé.

Puisque plusieurs points d'esquisse interviennent finalement, la zone où l'on pourra placer la source de lumière ponctuelle sera l'intersection des régions admissibles associées à chacun des points d'esquisse.

En essayant de maximiser ensuite la distance entre les points d'esquisse et la source lumineuse (tout en cantonnant celle-ci à la zone que l'on vient de spécifier précédemment), on parvient à affecter une position à la source d'éclairage ponctuelle. En maximisant la fonction précédente, la frontière de l'ombre produite est la plus proche possible de l'esquisse engendrée par l'utilisateur.

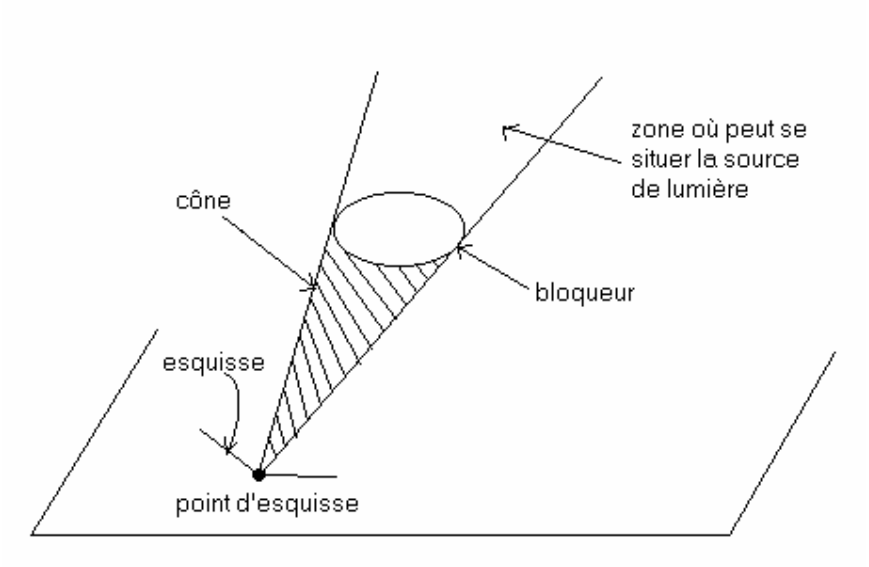


Figure 30. Détermination de la région valide pour le positionnement d'une source de lumière en fonction d'un point d'esquisse.

Le procédé que nous venons d'expliciter est mis en application sur la figure 31. Les trois premières images correspondent à respectivement 1 trait d'esquisse, 2 traits d'esquisses et enfin 3 traits d'esquisse. La source lumineuse ponctuelle calculée est représentée par un point blanc. La dernière image correspond à la même scène que celle ayant produit la troisième image, mais avec un point de vue différent.

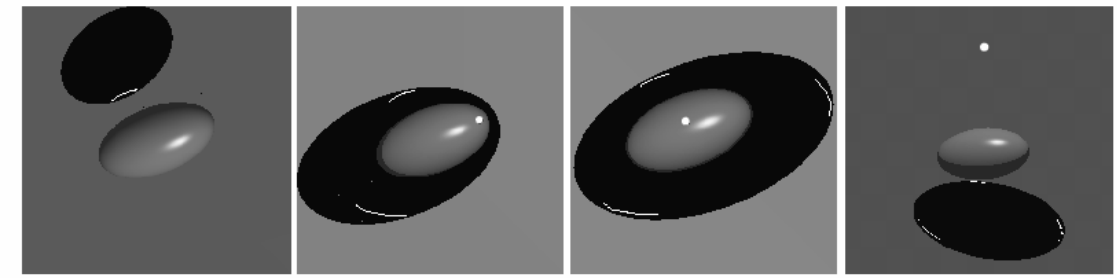


Figure 31. Création d'une source de lumière ponctuelle à partir d'esquisses de l'ombre produite par celle-ci (Images extraites de [PRJ97]).

γ) Gestion de sources lumineuses non réduites à un point à l'aide de zones d'ombre et de pénombre.

Les sources lumineuses non réduites à un point, comme les sources de lumière linéaires, polygonales, polyédriques, sphériques, etc. produisent trois types de régions :

1. Les régions totalement éclairées par la source lumineuse (Puisque l'on est dans le cas d'un éclairage, cela signifie que la région est totalement visible depuis la source de lumière).
2. Les régions situées complètement dans l'ombre (qui sont des régions totalement invisibles depuis la source lumineuse). Ce sont ces régions que nous avons traitées au α en recourant à des esquisses.

3. Les régions placées dans la pénombre (qui sont partiellement visibles depuis la source de lumière).

Traitement des régions situées dans l'ombre :

Il s'agit d'une modification assez simple du procédé que l'on a exposé au α.

On doit vérifier en supplément que tous les points de la source lumineuse se trouvent à l'intérieur de chaque zone convenable déterminée par chacun des cônes associés à chacun des points d'esquisse. On utilise ensuite un algorithme d'optimisation pour obtenir les sommets d'une source de lumière polygonale et convexe.

Traitement des régions situées dans la pénombre :

Les zones correspondant à la pénombre engendrée par un bloqueur sont modélisées avec des esquisses composées de points d'esquisse, de la même façon que l'on gère des zones d'ombre.

La condition que doit vérifier la source lumineuse est ici moins forte que celle liée aux régions placées dans l'ombre : seulement un point de la source de lumière doit être situé à l'intérieur de chaque zone convenable déterminée par chacun des cônes associés à chacun des points d'esquisse. On étudie pour cela l'intersection entre les cônes et la source lumineuse. Une technique d'optimisation permet ensuite de placer la source lumineuse en essayant de la rendre la plus éloignée possible du bloqueur.

La figure 32 montre le résultat d'une détermination d'une source de lumière rectangulaire. Les deux premières images correspondent à des esquisses d'ombre (Seul le point de vue change entre les deux images). Les deux dernières correspondent à des esquisses de pénombre.

Remarque :

Il y a certains problèmes d'éclairage inverse que l'on peut résoudre avec une source lumineuse ponctuelle mais qui deviennent insolubles quand on a recours à des sources de lumière qui sont ne sont pas ponctuelles (celles dont vient de parler auparavant).

Si l'on impose une taille assez importante à la source lumineuse et que l'on met celle-ci en présence d'un bloqueur de taille plus modeste, on ne pourra obtenir une ombre portée de taille plus grande que celle du bloqueur. Par contre, on pourra obtenir cette ombre portée si l'on se sert d'une source d'éclairage ponctuelle.

δ) Traitement de bloqueurs concaves et d'objets constitués de plusieurs bloqueurs élémentaires.

Dans le cas où le bloqueur considéré est concave, il est décomposé en objets convexes plus simples (ses composantes connexes). Nommons ces derniers des *bloqueurs élémentaires*.

On est ainsi conduit à examiner des objets formés à l'aide de plusieurs bloqueurs élémentaires.

Le logiciel issu des travaux de [PRJ97] permet de tracer une esquisse par bloqueur élémentaire. Il détermine alors une source lumineuse qui doit être placée dans l'intersection des volumes correspondant aux zones acceptables associées à chacun des points d'esquisse.

Par exemple, si l'on veut manipuler l'ombre produite par un arbre, on sélectionne une feuille (un bloqueur élémentaire) et l'on trace une esquisse. Puis on choisit une autre feuille et on ajoute une esquisse pour celle-ci, etc. A chaque esquisse introduite, on augmente les contraintes sur la position de la source lumineuse.

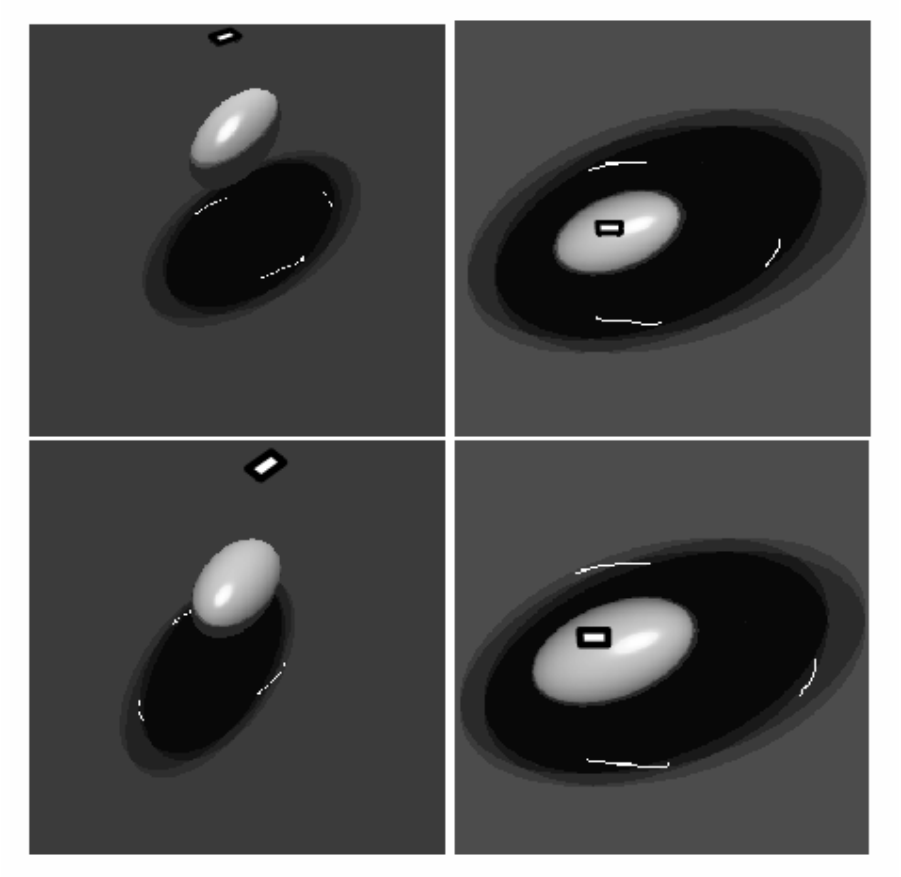


Figure 32. Détermination d'une source de lumière rectangulaire à partir d'esquisses d'ombre ou d'esquisses de pénombre (Images extraites de [PRJ97]).

La figure 33 met en valeur cette méthode. Trois esquisses d'ombres sont progressivement tracées et modifient à chaque fois la position d'une source lumineuse ponctuelle. Chaque esquisse est associée à un bloqueur élémentaire (Ici, les bloqueurs élémentaires sont les composantes connexes des diverses lettres de la scène).

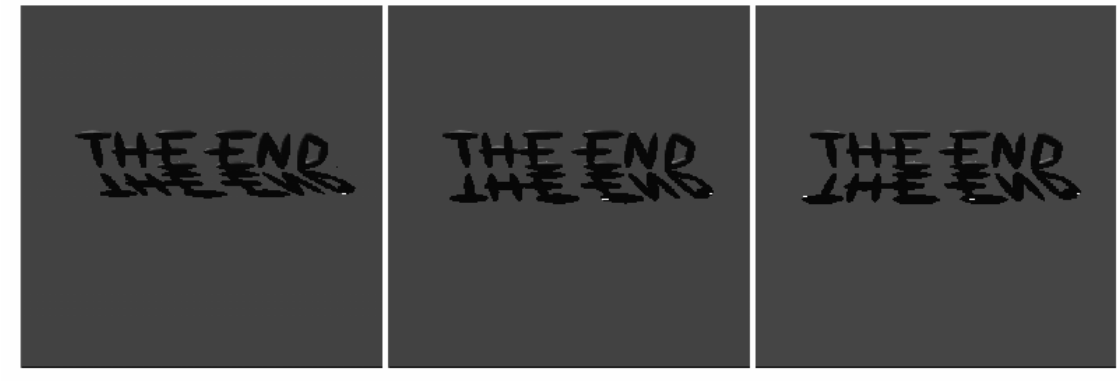


Figure 33. Détermination d'une source lumineuse ponctuelle avec un bloqueur non convexe (Images extraites de [PRJ97]).

c) Lumières obtenues à l'aide d'esquisses de zones fortement éclairées (*highlights*).

Dans cette partie de leur article, [PRJ97] essaient de déterminer la position d'une source lumineuse *ponctuelle* à partir du terme spéculaire intervenant dans le modèle de Phong. Nous allons donc reprendre les notations introduites au 2.5.2.1.a pour les paramètres apparaissant dans l'expression spéculaire.

L'utilisateur indique des zones fortement éclairées sur des surfaces de la scène à l'aide d'esquisses. Ces esquisses sont, comme dans le cas des zones d'ombre des lignes polygonales dont les points de contrôle sont aussi appelés points d'esquisse.

A chacun des points d'esquisse est associé un cône, et la source lumineuse se situe dans l'intersection d'un ensemble de cônes. Les cônes manipulés pouvant avoir une intersection vide, on modifie alors l'exposant spéculaire α à l'aide d'une valeur de seuil τ délimitant la zone de réflexion spéculaire (Les coefficients α et τ servent à caractériser les cônes). C'est encore un procédé d'optimisation qui affecte un emplacement à la source lumineuse : elle doit être la plus proche possible de l'axe du cône et la plus éloignée possible du sommet du cône.

La figure 34 présente deux esquisses de zones fortement éclairées, avec l'effet spéculaire produit quand la position de la source de lumière ponctuelle et l'exposant spéculaire associé au matériau ont été calculés.

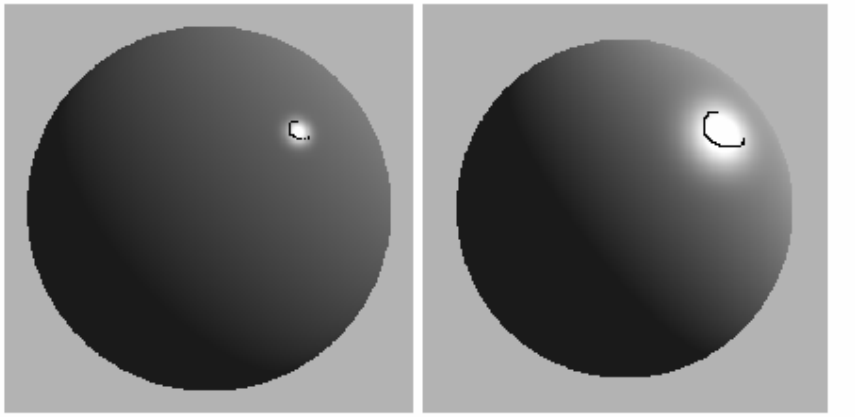


Figure 34. Exemples d'esquisses de zones fortement éclairées (Images extraites de [PRJ97]).

2.5.3 Articles se plaçant dans le cadre de la radiosité.

2.5.3.1 Introduction.

Le modèle de Phong ne s'appuie sur aucune théorie physique ; il est l'aboutissement de plusieurs techniques empiriques d'illumination. La radiosité est au contraire une technique de modélisation des échanges lumineux entre différentes parties d'une scène.

Pour le problème que nous étudions ici, l'un des grands intérêts de la radiosité est qu'elle est une méthode d'illumination globale. Ceci ne signifie pas que tous les articles faisant appel à la radiosité prennent en compte l'illumination indirecte des éléments constitutifs de la scène. Pour des raisons de rapidité de calcul, il arrive que les auteurs se limitent à un éclairage direct (parfois simplement pour les exemples illustrant la méthode proposée dans l'article).

Les deux articles fondamentaux dans ce domaine sont [SDSAG93] et [KPC93]. La plupart des études effectuées par la suite ne manquent pas de se situer par rapport à ceux-ci.

2.5.3.2 Détermination de l'émittance de sources lumineuses à partir de carreaux coloriés.

[SDSAG93] proposent un système dans lequel les positions des sources de lumière sont fixées à l'avance. L'utilisateur peint alors des zones de la scène, puis le logiciel essaie de déterminer les paramètres d'éclairage à affecter aux sources de lumière pour que l'image produite corresponde le mieux possible à ce que le concepteur vient de produire en coloriant certaines parties de la scène. Comme la réflectivité des carreaux de la scène est elle aussi fixée, on ne peut pas associer une couleur totalement arbitraire à ceux-ci quand on les peint avec l'interface mise à disposition. Pour conserver un bon degré d'interactivité, seule une illumination directe est mise en œuvre lors du rendu.

Afin de trouver l'intensité lumineuse des sources de lumière, [SDSAG93] ont recours à une technique d'optimisation linéaire. Il s'agit d'une méthode des moindres carrés. Examinons-là plus en détail :

Considérons $\{\Phi_1, \Phi_2, \ldots, \Phi_n\}$ l'ensemble des fonctions correspondant à l'influence des chacune des n sources de lumière de la scène. Comme l'on travaille dans le cadre de la radiosité, ces fonctions représentent la distribution de la radiance sur toutes les surfaces de l'environnement, radiance approchée par une combinaison linéaire de fonctions élémentaires construites à partir des radiosités calculées.

Soit Ψ la fonction correspondant à la scène coloriée par l'utilisateur.

On tente d'approcher Ψ par une combinaison linéaire des fonctions $\Phi_1, \Phi_2, \ldots,$ et Φ_n.

Cette combinaison linéaire Ω peut s'écrire sous la forme :

$$\Omega = \sum_{i=1}^{n} \omega_i \Phi_i$$

On doit alors minimiser la quantité $\|\Psi - \Omega\|$, où la norme est définie à partir d'un produit scalaire < , > sur un espace fonctionnel. Il s'agit d'un problème des moindres carrés dont la solution s'exprime sous la forme matricielle suivante :

$$Mw = b$$

Avec :

- $M = (<\Phi_i,\Phi_j>)_{i,j}$ est la matrice de Gram associée à la suite de fonctions $(\Phi_1, \Phi_2, \ldots ,\Phi_n)$
- w est le vecteur constitué par les coefficients que l'on recherche : les ω_i (qui correspondent à l'émittance dans le cas de la radiosité)
- b est le vecteur formé par les produits scalaires $<\Phi_i,\psi>$ pour i de 1 à n

Pour résoudre le système linéaire associé à l'équation matricielle précédente, [SDSAG93] emploient une méthode itérative : celle de Gauss-Seidel. Elle est cependant modifiée afin de ne pas obtenir des ω_i négatifs.

Une extension de l'algorithme est finalement incluse dans l'application développée par [SDSAG93] : elle consiste à s'appuyer sur les résultats obtenus pour une scène précédente lors des calculs accomplis après une modification par l'utilisateur d'une des couleurs d'un des carreaux de la scène. Les valeurs calculées précédemment servent comme valeur initiale pour l'approximation de w dans la méthode de Gauss-Seidel.

Remarque :

L'expression du problème d'éclairage inverse sous forme d'un problème d'optimisation très général permet d'appliquer l'algorithme à d'autres modèles d'illumination. Les fonctions $\Phi_1, \Phi_2, \ldots,$ et Φ_n peuvent ainsi correspondre à des images (pouvant être obtenues par un

algorithme de lancer de rayons), et le produit scalaire employé sera alors le produit scalaire usuel sur $\Re^p$ où p est le nombre de pixels des n images produites par Φ_1, Φ_2, …, et Φ_n. L'application du procédé à une gestion de la radiance de la scène est lui immédiat : il suffit de ne pas approcher la radiance par des fonctions linéaires ayant comme paramètres des valeurs de radiosité (approximation qui est accomplie dans le logiciel développé à partir des idées exposées dans l'article).

Sur la figure 35 sont représentées une spécification de l'éclairage fournie par un utilisateur (1ère image) et la meilleure approximation de celle-ci avec la méthode que nous venons d'expliquer (2ème image).

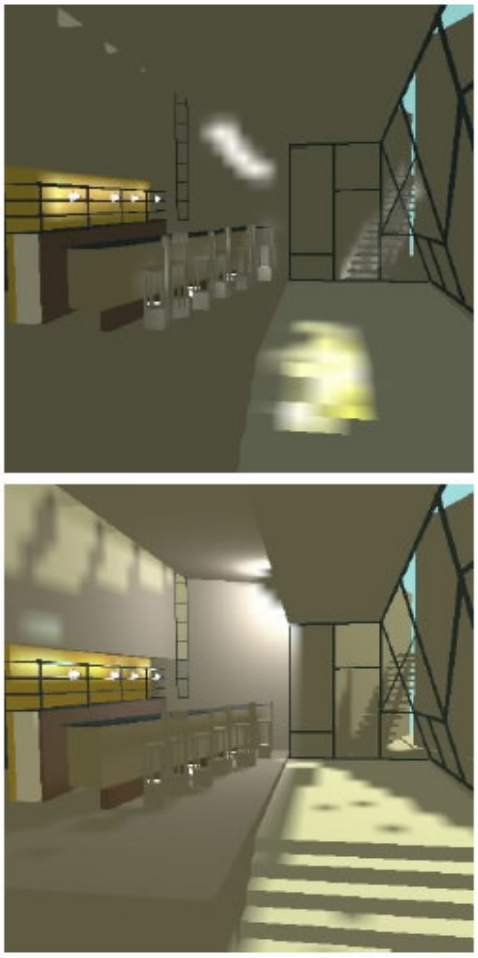


Figure 35. Calcul de l'émittance de certains carreaux fixés a priori, carreaux jouant le rôle de sources lumineuses et approchant le mieux possible une spécification de l'éclairage donnée par un utilisateur (Images issues de [SDSAG93]).

2.5.3.3 Radioptimisation : utilisation de la description d'un éclairage en terme d'ambiance et résolution à l'aide d'une technique d'optimisation.

a) Introduction.

[KPC93], recourant à la radiosité hiérarchique [HSA91], fournissent à l'utilisateur un outil calculant les caractéristiques physiques des sources lumineuses et des autres éléments de la

scène, à partir d'une description de l'éclairage pouvant se présenter sous forme qualitative (La pièce peut donner une impression de « clarté », d' « intimité », ou paraître « agréable »).

Les positions des sources de lumière étant fixées, un algorithme d'optimisation, celui de Broyden-Fletcher-Goldfarb-Shanno (qui appartient à la famille des méthodes de Quasi-Newton), détermine selon la fonction objectif construite par l'utilisateur :

- L'émittance E_i des sources lumineuses idéalement diffuses.
- Les paramètres des spots (introduits dans le contexte de la radiosité en modifiant l'expression du facteur de forme) : la direction V_i du spot et l'exposant n_i contrôlant la concentration de la lumière autour de l'axe du cône du projecteur.
- Les réflectivités ρ_i de certains éléments de la scène.

Nous allons étudier la formulation du problème d'optimisation, les variables qui interviennent, la gestion des contraintes, et les possibilités de fabrication de la fonction objectif par l'utilisateur.

b) Les variables du problème d'optimisation.

Nous les avons pour la plupart déjà énumérées lors du paragraphe introductif. On doit y ajouter les radiosités B_i de certains éléments. Ces valeurs B_i sont calculées par l'algorithme de résolution de la radiosité hiérarchique.

On n'a en pratique qu'un nombre restreint de variables à déterminer (appelées « variables libres »), car l'on n'a pas à rechercher E_i, V_i, n_i et ρ_i pour chaque élément i.

c) Les contraintes du problème d'optimisation.

[KPC93] distinguent trois types de contraintes :

1. Les contraintes physiques issues de l'équation de radiosité.
2. Les contraintes correspondant aux désirs du concepteur, qui peuvent être des égalités ou des inégalités portant sur un ou plusieurs éléments de la scène. Par exemple, l'utilisateur peut vouloir que la radiosité d'un élément soit bornée.
3. Les contraintes ayant un rôle de barrières, qui servent à garantir que le modèle est physiquement correct. Ainsi, la réflectivité de chaque élément doit être comprise entre 0 et 1.

Il y a une ressemblance dans la forme des contraintes introduites par l'utilisateur et les contraintes assurant la cohérence du modèle physique. Cependant, les premières sont des desiderata qui ne seront pas forcément parfaitement satisfaits par le processus d'optimisation, alors que les dernières doivent absolument être respectées.

d) La fonction objectif du problème d'optimisation.

La fonction d'objectif est une combinaison linéaire de diverses fonctions, combinaison choisie par l'utilisateur du logiciel. Le concepteur souhaitant un certain type d'éclairage va ajuster en conséquence les poids associés à chacune des fonctions de base intervenant dans la fonction objectif.

Détaillons les diverses fonctions objectifs de base mises à disposition :

- L'énergie totale de la scène (fonction servant à contrôler la consommation d'énergie).
- Des fonctions liées à la brillance (*brightness*) de la pièce. Comme l'énergie totale de la pièce, ces fonctions correspondent à des paramètres physiques.

- Des fonctions se rapportant à une impression de « clarté », ou d' « intimité », ou d'un aspect « agréable » pour l'utilisateur. Les caractéristiques de ces fonctions ont été obtenues à la suite de tests pendant lesquels ont été présentées à des individus plusieurs images d'une même scène avec différents types d'éclairage. Ces tests ont permis d'établir une corrélation entre les perceptions subjectives de « clarté », etc. et les fonctions modélisant la brillance de la scène.
- Des fonctions issues des contraintes. En effet, la technique d'optimisation employée est non contrainte. Pour éliminer les contraintes, on ajoute à la fonction objectif des fonctions pénalisantes c'est-à-dire des fonctions qui prendront une valeur élevée si les contraintes ne sont pas respectées par les valeurs des variables libres. C'est une méthode classique pour passer d'un problème d'optimisation avec contraintes à un problème d'optimisation sans contraintes.

On peut ainsi écrire la fonction objectif F sous la forme suivante :

F = termes physiques + termes basés sur une perception subjective

e) La phase d'optimisation.

Une fois que l'utilisateur a créé la fonction objectif F, un algorithme de Quasi-Newton essaie de minimiser F (On minimise puisque l'on a des fonctions pénalisantes).

La méthode de Newton recherche des extremums locaux de façon itérative. A partir de la position courante dans l'espace des variables libres parcouru, on détermine une direction et un pas qui vont fournir la nouvelle position, avec convergence du « chemin » accompli vers un extremum. L'algorithme suivant présente une formulation classique de la méthode de Newton :

Objectif : Obtenir un minimum local d'une fonction numérique g.

Données :

- Un point initial $x \in \Re^n$;
- Une valeur seuil $\varepsilon > 0$.

Algorithme :

Tant que $\|\nabla g(x)\| > \varepsilon$

Calculer v tel que $\nabla^2 g(x)v = -\nabla g(x)$

$x \leftarrow x + v$

Fin Tant que

$\nabla g(x)$ est le gradient de g en x. $\nabla^2 g(x)$ est la matrice hessienne (matrice des dérivées partielles secondes) de g en x. Le vecteur v est la nouvelle direction calculée à chaque pas de l'algorithme.

Les méthodes de Quasi-Newton ont l'avantage sur la méthode de Newton de ne pas calculer la matrice hessienne de g qui sert à trouver la nouvelle position, mais simplement d'en effectuer une approximation moins coûteuse.

On notera que pour le calcul des dérivées partielles premières (nécessaire pour les méthodes de Newton et celles de Quasi-Newton), [KPC93] tirent parti de la structure mise en œuvre pour la radiosité hiérarchique.

f) Conclusion.

[KPC93] est le seul article, à notre connaissance, prenant en compte une description qualitative de l'éclairage désiré. C'est une première étape pour une caractérisation de l'ambiance d'une scène.

Deux exemples de configurations lumineuses obtenues avec la méthode de [KPC93] sont montrés sur la figure 36. Les exemples correspondent à la même scène (au point de vue géométrique) ; seuls les souhaits du concepteur différent. Dans le premier exemple, l'utilisateur a demandé une impression de clarté ; dans le second, une atmosphère intimiste.

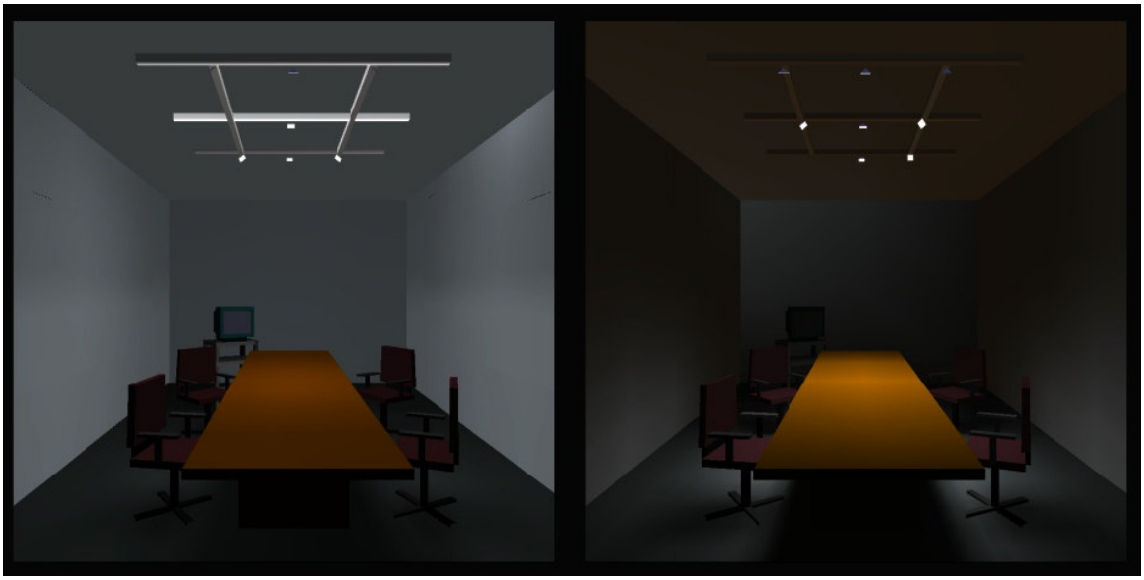


Figure 36. Deux éclairages d'une même pièce correspondant à deux spécifications différentes. L'image de gauche doit produire une impression de clarté. L'image de droite doit restituer une impression d'intimité. (Images extraites de [SDSAG93]).

En examinant la seconde scène, nous avons quelques réticences à qualifier son ambiance d'intimiste. En effet, la configuration de la scène (avec seulement une grande table, quelques chaises et un ensemble vidéo au fond de la pièce) est caractéristique d'un lieu de travail. En introduisant une lumière tamisée, on n'aboutit finalement qu'à une scène assez étrange.

Il y a une ambiance intrinsèque aux divers constituants d'une scène. L'ajout d'un éclairage censé donner une ambiance différente à celle portée par les éléments de la scène conduit à la génération d'images sémantiquement incohérentes. L'éclairage à lui seul ne saurait engendrer une ambiance particulière pour une scène quelconque. Ce n'est que si les éléments de la scène ne s'opposent pas à l'ambiance désirée que la méthode de [KPC93] peut déterminer un éclairage qui correspondra assez bien aux souhaits du concepteur en matière d'ambiance.

Par ailleurs, l'interface proposée par [KPC93] se révèle peu intuitive, puisque l'on doit parfois entrer des valeurs numériques pour les contraintes sur les propriétés physiques des éléments ou pour les poids des fonctions objectifs de base. Avec la technique d'optimisation employée, on est même parfois amené à rajouter des contraintes pour éviter des minima locaux ne présentant aucun intérêt pour le concepteur.

2.5.3.4 Utilisation d'une technique d'algèbre linéaire : la pseudo-inversion à l'aide d'une décomposition par valeurs singulières.

Dans [CM97], les auteurs répartissent les différents carreaux de la scène en 3 ensembles :

1. n_1 carreaux sont des sources potentielles de lumière (Ils ont une émittance qui peut être non nulle).
2. n_2 carreaux ont une émittance nulle et l'on n'a aucune information ou contrainte sur leur radiosité. Pour ces n_2 carreaux, on ne connaît donc a priori que leur géométrie et leur position dans la scène. L'algorithme d'éclairage inverse (ici un procédé mathématique d'algèbre linéaire) leur attribuera une radiosité en fonction des contraintes introduites à l'aide de la troisième catégorie de carreaux.
3. n_3 carreaux ont une émittance nulle et leur radiosité est fixée.

Le système d'équation de la radiosité s'écrit alors en tenant compte des 3 classes de carreaux précédentes :

$$\begin{pmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} e_1 \\ 0 \\ 0 \end{pmatrix}$$

où :

- les $(M_{ij})_{ij}$ sont des blocs correspondant à des caractéristiques uniquement géométriques de la scène.
- b_1, b_2, et b_3 sont 3 vecteurs contenant les radiosités des carreaux appartenant aux 3 classes que l'on a explicitées précédemment.
- e_1 est un vecteur contenant les n_1 valeurs d'émittance des carreaux pouvant être des sources lumineuses.

En notant n le nombre total de carreaux de la scène (avec $n = n_1 + n_2 + n_3$), on observe qu'il y a :

- n équations.
- $2n_1 + n_2$ inconnues (les valeurs des vecteurs e_1, b_1 et b_2).

On peut alors réécrire l'équation sous la forme matricielle suivante :

$$\begin{pmatrix} I & M_{11} & M_{12} \\ 0 & -M_{21} & M_{22} \\ 0 & -M_{31} & M_{32} \end{pmatrix} \begin{pmatrix} e_1 \\ b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} M_{13}b_3 \\ M_{23}b_3 \\ M_{33}b_3 \end{pmatrix}$$

Soit encore :

$$M \begin{pmatrix} e_1 \\ b_1 \\ b_2 \end{pmatrix} = N$$

Remarque : Si $e_1^i \neq 0$, le carreau i est une source de lumière et on a l'égalité : $b_1^i = e_1^i$.

Pour déterminer e_1, b_1 et b_2, [CM97] préconisent d'effectuer une pseudo-inversion de M en utilisant sa décomposition par valeurs singulières.

La décomposition en valeurs singulières de la matrice M consiste à écrire celle-ci sous la forme $M = US^tV$ où :

- U et V sont des matrices orthogonales,
- S est une matrice diagonale.

Les valeurs sur la diagonale de S sont appelées les valeurs singulières de M. Les valeurs singulières de M sont les racines carrées des valeurs propres de la matrice ${}^tMM$.

La pseudo-inverse de la matrice M est la matrice M^+ définie par la relation suivante :

$$M^+ = V S^+ {}^tU$$

où :

- U et V sont les matrices orthogonales apparaissants dans la décomposition en valeurs singulières de M,
- S^+ est la matrice obtenue en transposant S et en remplaçant les valeurs non nulles par leurs inverses.

Pour obtenir les valeurs e_1, b_1 et b_2, on calcule finalement le produit M^+N. Cette méthode de calcul fournit la solution optimale (au sens des moindres carrés) à l'équation formulée par [CM97].

Remarque :

De par la nature des entités mathématiques qu'il manipule, l'algorithme utilisé semble assez onéreux en terme de temps de calcul. Ainsi l'exemple illustratif de l'article est-il une scène constituée uniquement de 228 carreaux, où l'on n'introduit pas de carreaux de la 2ème catégorie. La scène employée pour valider la méthode est présentée sur la figure 37.

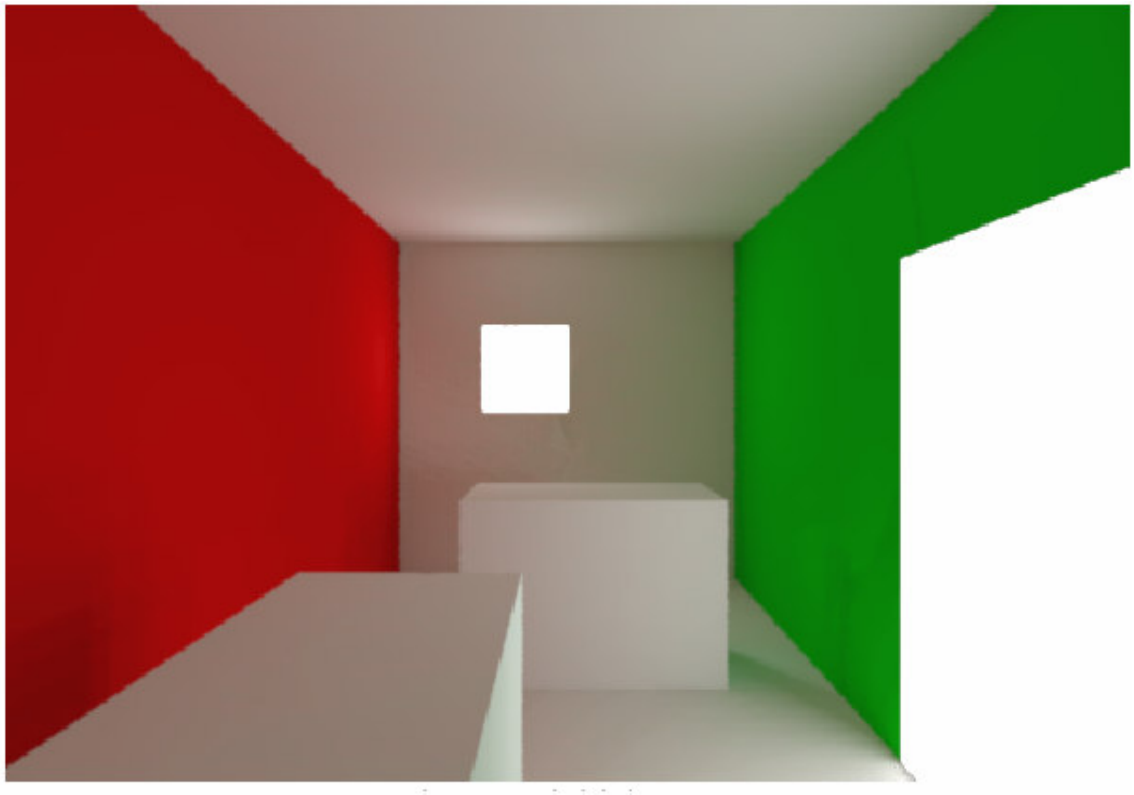


Figure 37. Scène de test utilisée dans [CM97] pour prouver la validité de la méthode proposée.

La scène test contient deux sources de lumière. A l'aide de Radiance [WAR94] les radiosités des divers carreaux de la scène sont déterminées.

Pour la mise en œuvre de l'algorithme d'éclairage inverse, [CM97] choisissent comme sources lumineuses potentielles tous les carreaux de la face du fond et de celle de droite. Ils affectent aux autres carreaux de la scène les radiosités calculées précédemment avec Radiance. Une fois la méthode appliquée, les émittances trouvées sont très proches de celle de la scène test initiale.

On remarquera que [CM97] n'ont fait que prouver l'exactitude mathématique de leur méthode dans un cas où le système d'équations à résoudre est sur-contraint (puisque $n_3 > n_1$).

[CON02a] et [CON02b] proposent une étude plus approfondie ainsi qu'une amélioration de la méthode introduite dans [CM97]. Les grandes lignes de la méthode demeurent puisque c'est toujours le même système d'équations que l'on résout avec une pseudo-inversion de matrice à l'aide d'une décomposition en valeurs singulières.

Alors que [CM97] ne permettait pas à l'utilisateur de spécifier ses choix en matière d'éclairage, une interface a depuis été développée. Elle permet d'assigner des radiosités à certains carreaux de la scène (carreaux qui correspondent au troisième genre de carreaux de [CM97]) et d'interdire à certains carreaux d'être des sources de lumière (carreaux qui correspondent au deuxième genre de [CM97] dont l'émittance est nulle).

[CON02b] accorde une attention particulière aux valeurs singulières proches de 0. Ces valeurs peuvent être non nulles à cause d'approximations incorrectes dans les calculs. Dans la pseudo-inversion, comme on fait intervenir l'inverse des valeurs singulières, ces quantités deviendront très grandes alors qu'elles devraient être nulles. Pour pallier à ce problème, [CON02b] choisit de mettre à 0 toutes les valeurs singulières au-dessous d'un certain seuil dépendant de la précision du type de données manipulé lors des calculs numériques.

En analysant les résultats obtenus dans les cas où le système à traiter n'est pas sur-contraint, on remarque que :

- Des valeurs négatives pour les émittances peuvent intervenir (valeurs qui n'ont bien évidemment aucun sens d'un point de vue physique).
- La méthode a tendance à créer des configurations d'éclairage irréalistes (beaucoup de carreaux avec une émittance non nulle dont l'ensemble ne constitue pas une forme précise).

Pour remédier à ces difficultés, [CON02b] met en place un processus en deux étapes :

1. On élimine des carreaux jusqu'à obtenir une solution physiquement valide (c'est-à-dire avec des émittances positives). Quand on supprime une source lumineuse avec une émittance négative, cette émittance est redistribuée aux carreaux émetteurs voisins.
2. Ensuite, on continue à éliminer des carreaux émetteurs jusqu'à ce que l'écart entre les radiosités obtenues et celles voulues soit au-dessous d'un certain seuil, ou jusqu'à ce que l'on atteigne un nombre de sources lumineuses minimal (nombre que l'utilisateur peut fixer).

Sur la figure 38, six solutions d'un problème inverse sont présentées.

La scène étudiée est constituée d'approximativement 1000 carreaux. Les carreaux pouvant posséder une émittance non nulle sont ceux du plafond et du mur de gauche. L'image (a) correspond à la scène obtenue avec la pseudo-inversion. L'image (d) où se trouvent 14 sources de lumière est la première solution physiquement correcte déterminée par l'algorithme. Les images suivantes correspondent à un nombre de sources lumineuses choisi par l'utilisateur. On remarquera qu'avec un nombre de carreaux potentiellement émetteurs supérieurs à 7, la scène présente un éclairage peu convaincant (Les formes constituées par l'agrégation des carreaux émetteurs ne sont pas conventionnelles).

Pour la scène, [CON02b] indique 29 minutes comme temps de calcul pour la totalité du processus d'inversion avec un Pentium III à 450 Mhz.

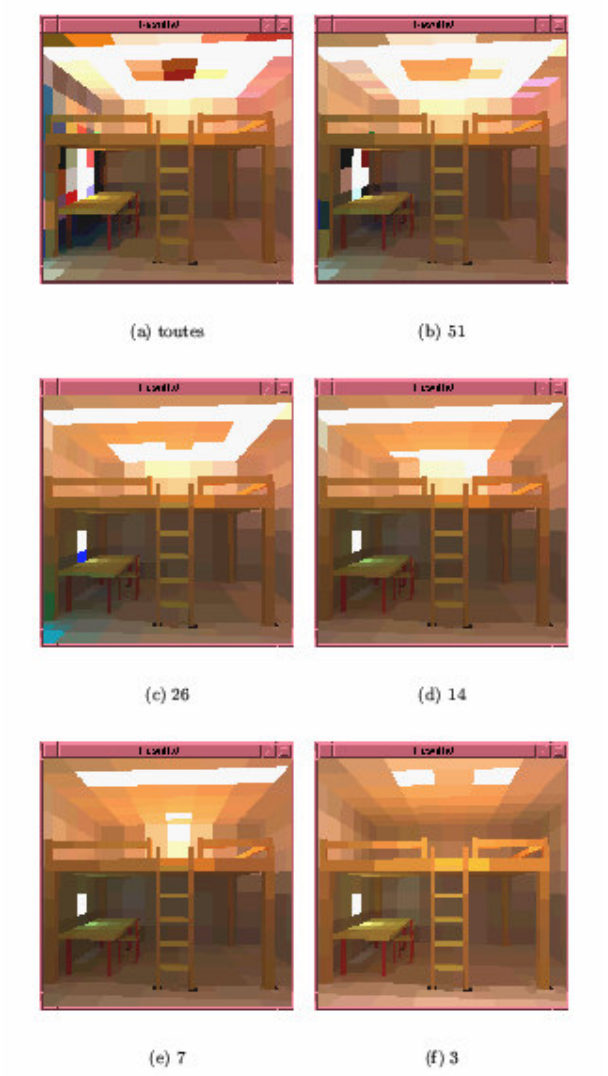


Figure 38. Les différentes scènes produites par l'algorithme de [CON02b] jusqu'à la réduction du nombre de carreaux émetteurs à 3.

Une approche très similaire à celle de [CM97] est développée dans [HMH95], quoique dans le domaine des transferts d'énergie thermiques. En replaçant la problématique dans le cadre de la radiosité, il s'agit, à partir de radiosités fournies pour certaines surfaces de la scène, de trouver l'émittance d'un ensemble de surfaces (pas nécessairement distinctes de celles dont on a précisé la radiosité). Après avoir subdivisé les surfaces en carreaux assez petits pour les considérer comme des éléments différentiels, les auteurs procèdent aussi à une pseudo-inversion de matrice en utilisant une forme de décomposition par valeurs singulières.

On notera toutefois que les essais accomplis sont limités à la 2D.

2.5.3.5 Utilisation d'un algorithme génétique.

[ER97] abordent le problème d'éclairage inverse de la même façon que [SDSAG93] : l'utilisateur affecte des couleurs à différents carreaux de la scène, puis un algorithme détermine le nombre, la position et l'intensité lumineuse de sources lumineuses de telle sorte que la scène obtenue avec ces sources d'éclairage soit la plus proche possible de la scène « coloriée » par l'utilisateur.

Comme méthode de calcul de la radiosité, [ER97] utilisent la radiosité hiérarchique [HSA91].

L'algorithme employé pour résoudre le problème est un algorithme génétique [GOL94] dont seulement quelques spécifications sont données :

- Un individu est une chaîne binaire représentant le nombre de lumières, et, pour chacune de celles-ci, leur émittance, leur position et leur type (La nature de ce « type » n'est pas précisée dans l'article).
- Pour déterminer l'importance d'un individu, on mesure l'erreur RMS entre la scène désirée par le concepteur et celle que l'on aurait en se servant des paramètres encodés par l'individu. Le fait d'utiliser la radiosité hiérarchique permet ainsi d'accélérer les calculs de la fonction de sélection.

Remarque :

Comme dans la méthode exposée au chapitre précédent, les scènes testées sont de taille modeste (de l'ordre de 100 à 500 éléments dans les structures d'arbre quaternaire de la radiosité hiérarchique). Là aussi, les temps de calcul paraissent prohibitifs.

La figure 39 présente une scène où l'utilisateur a peint certains carreaux (image de gauche) et l'éclairage obtenu avec la méthode permettant d'approcher de la scène voulue par l'utilisateur (image de droite).

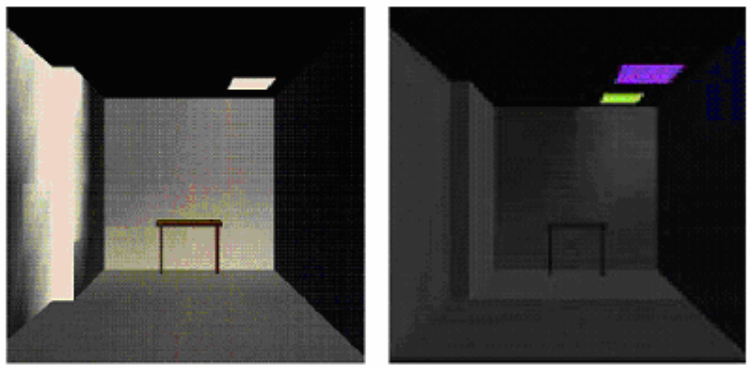


Figure 39. Une scène objectif fournie par l'utilisateur (image de gauche) et son approximation par l'algorithme de [ER97] (image de droite).

Les résultats présentés par [ER97] ne semblent pas visuellement convaincants.

2.5.3.6 Utilisation d'une méthode de Monte-Carlo.

Les travaux de [OH95], issus du domaine des échanges d'énergie calorique, font appel à la technique probabiliste de Monte-Carlo. Nous retraduisons ici leur problématique en terme de radiosité.

Dans la conception du problème inverse choisie, [OH95] essayent de trouver l'émittance de certains carreaux de la scène sachant que l'on connaît la radiosité d'autres carreaux (Ces deux ensembles de carreaux étant disjoints).

Depuis les carreaux dont la radiosité est fixée, on lance aléatoirement des rayons dans la scène (à l'aide d'un hémisphère subdivisé régulièrement pour calculer la direction des rayons). Dans l'article, seuls les rayons reliant directement deux carreaux sont pris en compte, mais il est possible de généraliser l'algorithme afin qu'il utilise plusieurs réflexions sur les éléments de la scène. Chaque rayon transporte une certaine quantité d'énergie que

l'algorithme lui attribue. Pour chaque carreau dont on tente de déterminer l'émittance, on effectue une somme pondérée de toutes les quantités d'énergies fournies par les rayons qui atteignent le carreau.

Remarque :
La technique que l'on vient de décrire n'a été testée que pour des environnements en 2D.

2.5.4 Article se plaçant dans le cadre d'une évaluation de la radiance.

2.5.4.1 Introduction.

Dans [CSF99a] et [CSF99b] est présenté un logiciel permettant de placer différentes sources lumineuses répondant à diverses contraintes fournies par l'utilisateur. Le système décrit prend en compte une illumination globale et a recours à une technique d'optimisation pour résoudre le problème d'éclairage inverse.

On peut décomposer l'application en plusieurs unités fonctionnelles :

- un analyseur de scripts que l'utilisateur écrit, scripts servant à spécifier les contraintes pour l'éclairage de la scène et utilisant un langage particulier.
- Radiance [WAR94], qui calcule la radiance en un point dans une direction donnée.
- ASA [ING98], un paquetage résolvant les problèmes d'optimisation à l'aide d'une technique de recuit simulé.

2.5.4.2 Les hypothèses concernant le modèle physique du logiciel.

Certaines hypothèses sont assumées pour l'implémentation :

- Le milieu est non participant.
- La BRDF est symétrique (c'est-à-dire qu'il n'y a aucun changement si l'on intervertit la directions des rayons lumineux). La BRDF (*Bidirectional Reflectance Distribution Function*) caractérise la façon dont une surface réfléchit l'énergie selon son orientation.

La seconde hypothèse permet la *réversibilité des calculs* concernant l'éclairage. Il s'agit d'une condition très importante vu la façon dont sont modélisés les objectifs d'éclairage (cf. le chapitre suivant).

2.5.4.3 Les différents types de sources lumineuses manipulées par le logiciel.

Les types dont il est ici question ne concernent pas la nature mais le rôle que joue une source lumineuse dans le système.

La modélisation proposée distingue 3 sortes de sources de lumière (Nous avons conservé les abréviations issues des termes anglais pour pouvoir se référer plus aisément à l'article) :

- Les PL (*Previous Luminaire*) qui correspondent à des sources lumineuses déjà présentes dans la scène.
- Les IL (*Inverse Luminaire*). Ce sont des sources lumineuses *fictives* qui servent à modéliser les objectifs que le concepteur a en vue. Par exemple, si l'on désire qu'une table soit bien éclairée, il y aura une IL associée à cette table et qui répandra de l'énergie lumineuse dans la scène. On pourra ensuite évaluer l'impact de la radiance émise par l'IL sur une position dans la scène. C'est à l'aide de l'importance d'une IL pour un point de la scène que l'on déterminera la position de la source lumineuse (une DL, cf. le troisième type de source lumineuse) qui créera l'ambiance voulue. On remarque ici l'importance de la réversibilité des calculs,

puisque l'algorithme essaie de trouver les sources d'éclairage en émettant des rayons lumineux depuis les endroits où l'on souhaite un éclairage particulier.

- Les DL (*Desired Luminaire*) qui sont les sources de lumière qui vont produire l'éclairage voulu.

2.5.4.4 La description par le concepteur des contraintes et des objectifs à atteindre.

L'utilisateur spécifie l'éclairage qu'il désire à l'aide d'un script qui correspondra à la fonction de coût dans la phase d'optimisation. Nous allons étudier cette étape du processus sur un exemple.

Considérons une pièce contenant une table et éventuellement des sources lumineuses (des PL pour reprendre la terminologie introduite au 2.5.4.3). On souhaite améliorer l'éclairage de la pièce en rajoutant un projecteur de telle sorte que l'éclairage de la table soit augmenté, mais sans qu'une personne assise devant la table soit éblouie par la lumière du projecteur.

Dans ce cas, nous avons deux IL : un lié à l'illumination de la table et un autre gérant le fait que la personne ne soit pas éblouie (On assimile la figure de la personne à une facette fictive de la scène).

Le script correspondant à cette description des objectifs, programme que l'utilisateur doit entièrement fabriquer, est le suivant :

1 # (x1,x2,x3) : position de la DL, (x4,x5) : direction du spot (de la DL)
2 V = Vector($Face_{Center}$, (x1,x2,x3)) # Face désigne le visage de la personne
3 If Angle($Face_{Perp}$,V) < $V_{Threshold}$ and
4 Angle($Face_{Perp}$,Dir(-x4,-x5))<$A_{Threshold}$ return FAILURE
5 $W_{IL1/DL1}$ = Importance ($SCENE_{IL1}$,x1,x2,x3,x4,x5)
6 $W_{IL2/DL1}$ = Importance ($SCENE_{IL2}$,x1,x2,x3,x4,x5)
7 Return –(K1* $W_{IL1/DL1}$ – K2* $W_{IL2/DL1}$)

Commentaires explicatifs :

Les lignes 3 et 4 permettent d'introduire des contraintes sur les paramètres caractérisant la source lumineuse. La condition de la ligne 3 permet d'éviter que la source lumineuse ne se trouve en face de la personne placée devant la table. La condition de la ligne 4 interdit à l'axe du spot de se trouver dans la direction du visage de la personne.

Les lignes 5 et 6 servent à mesurer l'impact des IL sur la position (x1,x2,x3) et la direction (x4,x5) des sources lumineuses à placer dans la scène. La fonction Importance calcule la radiance à l'endroit et selon la direction précisés.

La ligne 7 renvoie le résultat de la fonction coût pour les paramètres (x1,x2,x3,x4,x5). On a une pondération des contributions des IL avec les coefficients K1 et K2. La différence de signe entre les IL pour la contribution à la fonction coût provient du fait que, pour la table, on a un maximum à atteindre (On veut que la table soit bien éclairée), alors que pour la figure de la personne, on a un minimum à ne pas dépasser (On veut que la personne ne soit pas éblouie).

2.5.4.5 Gestion de la radiance selon les objectifs.

Les IL qui modélisent les objectifs que le concepteur souhaite voir atteints propagent de l'énergie lumineuse dans la scène. Examinons comment s'effectue le calcul de la radiance que ceux-ci (les IL) doivent émettre.

Préalablement, on évalue l'influence des PL (qui sont, rappelons-le, des sources lumineuses placées initialement dans la scène) sur les IL. On peut détecter ainsi si l'on aboutit

à une contradiction (Par exemple, on désire qu'une zone soit peu éclairée, mais il y a déjà dans la scène une source de lumière qui l'illumine fortement). Selon l'impact qu'ont les PL sur une IL, on peut subdiviser celle-ci en plusieurs parties – et créer de la sorte de nouvelles IL – où la radiance sera pratiquement constante sur chacune des parties. Une fois la subdivision des PL éventuellement accomplie, on détermine la radiance qu'il faut distribuer dans une direction pour que les objectifs soient remplis. C'est cette radiance due à une source lumineuse fictive (une IL) qui aura un impact sur la fonction Importance employée dans un script.

2.5.4.6 Fonctionnement général de l'algorithme.

Nous avons vu que le concepteur exprimait ses désirs en programmant un script dont nous avons détaillé les principales spécifications. Le logiciel détermine ensuite la radiance que doit émettre chaque IL (source lumineuse fictive représentant l'un des objectifs de l'utilisateur).

Nous abordons ensuite la phase d'optimisation. Elle doit mener à la détermination d'une ou plusieurs sources lumineuses satisfaisant les exigences du concepteur.

[CSF99a] et [CSF99b] utilisent un algorithme de recuit simulé comme technique d'optimisation. Ils ont recours pour cela au paquetage ASA qui implémente un algorithme de ce type.

Nous allons exposer brièvement ici les grands principes de la méthode du recuit simulé (dont on pourra trouver une présentation plus détaillée dans [PTVF92]).

Le problème consiste à minimiser une fonction f non convexe sur $\Re^n$.

Dans ce cas, on a fréquemment recours à une méthode de « descente ». Partant d'un point $x_0 \in \Re^n$, on construit une suite de points dont les termes se rapprochent progressivement du minimum global recherché. Classiquement, la descente s'effectue suivant le gradient de f, avec une formule de récurrence du type :

$$x_{n+1} = x_n + r_n \nabla f(x_n)$$

où r_n est le pas, dépendant de la version de l'algorithme employée.

L'inconvénient de cette approche est que l'on risque d'aboutir à un minimum local.

Le recuit simulé tente de remédier à cette impasse en permettant à la suite $(x_n)_n$ de s'extraire d'un minimum local. Pour cela, on ajoute au second membre de l'équation de récurrence une composante aléatoire telle que $f(x_{n+1}) > f(x_n)$ avec une probabilité $p_n \to 0$ quand $n \to +\infty$. Cette modification du schéma classique de la descente offre au point x_n la possibilité (théorique) d'explorer tout l'espace avant de se mettre à descendre vers un minimum (Ceci dépend de la méthode choisie pour faire décroître la suite $(p_n)_n$ au cours du temps).

Le calcul de la radiance en un point et pour une direction donnée se fait à l'aide du logiciel Radiance. C'est Radiance qui trouve l'importance d'un IL pour les paramètres de la fonction coût.

Une fois que les sources lumineuses remplissant les exigences de l'utilisateur en matière d'éclairage ont été trouvées, Radiance procède à un rendu de la scène solution.

[CSF99b] proposent une accélération du calcul de la radiance. Pour cela, ils tiennent compte des calculs effectués préalablement autour du point analysé (Puisque l'algorithme d'optimisation fait évoluer celui-ci jusqu'à la meilleure position possible).

2.5.4.7 Traitement d'une source lumineuse non ponctuelle.

Pour introduire des sources lumineuses qui ne soient pas uniquement ponctuelles, [CSF99a] et [CSF99b] procèdent à un échantillonnage sur la source de lumière d'un autre type. Par exemple, pour une source lumineuse rectangulaire, ils choisissent 4 points à l'intérieur de celle-ci.

2.5.4.8 Exemples de résultats obtenus.

Le premier exemple, tiré de [CSF99a] illustre l'éclairage de deux bureaux symbolisés par des polygones. La configuration géométrique de la pièce apparaît sur la figure 40.

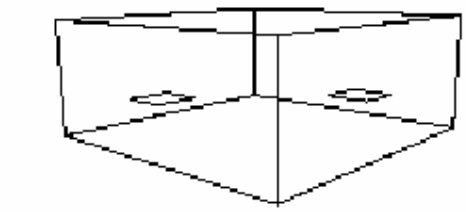


Figure 40. Pièce contenant deux bureaux (modélisés par des polygones) que l'on souhaite éclairer avec deux sources de lumière (Image extraite de [CSF99a]).

L'utilisateur désire avoir un éclairage homogène pour chacun des bureaux à l'aide de deux sources (Son idée est de ne pas de avoir de bureau partiellement éclairé). Après avoir défini un script correspondant à son souhait, il obtient deux solutions correspondant aux résultats de la phase d'optimisation. Ces solutions sont présentées sur la figure 41.

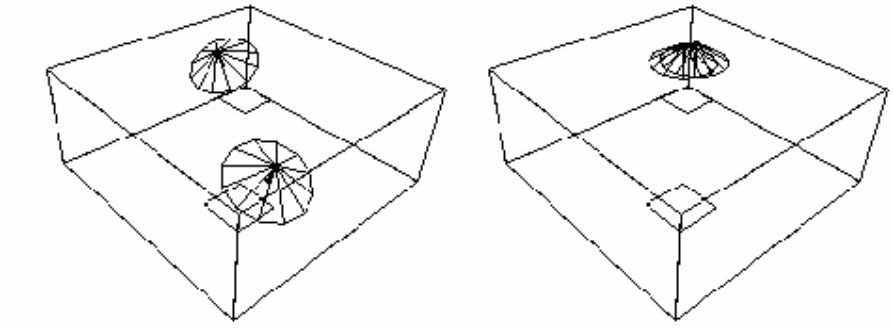


Figure 41. Les deux solutions obtenues pour une spécification à l'aide d'un script mal programmé (Image extraite de [CSF99a]).

On peut penser que la deuxième solution fournie par le logiciel ne correspond pas à ce qu'attendait le concepteur. Ceci provient du fait qu'il n'a pas écrit un script suffisamment précis pour écarter la solution où les deux sources lumineuses sont concentrées sur un seul bureau. On voit donc que l'écriture d'un script traduisant exactement les désirs de l'utilisateur n'est pas évidente.

Le deuxième exemple, issu celui-ci de [CSF99b], fait intervenir une scène plus complexe, constituée de 10550 objets (dont la plus grande partie sont des polygones) et représente une salle de travail de 6 boxs dans une SSII. L'éclairage de la pièce doit respecter les contraintes suivantes :

- Les programmeurs ne doivent pas être éblouis quand ils sont devant leur machine.

- L'écran, le clavier et la surface de travail d'un programmeur ne doivent pas être dans la pénombre.
- Les sources lumineuses doivent être alignées.

La figure 42 montre une solution optimale à ce problème d'éclairage calculée par le logiciel. Dans ce cas, on a optimisé les positions et les orientations de 3 spots.

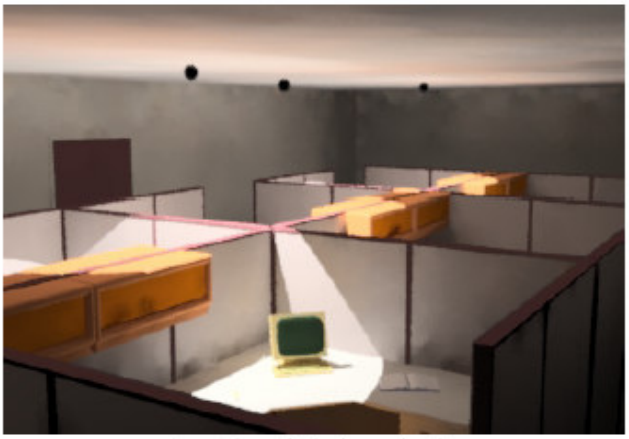


Figure 42. Eclairage d'une salle de travail informatique avec 3 spots (Image extraite de [CSF99b]).

Il a fallu environ 3 heures pour arriver à cette configuration d'éclairage en utilisant la technique permettant d'accélérer le calcul de la radiance (technique exposée dans [CSF99b]).

2.5.4.9 Conclusion.

Les exemples illustratifs de l'article sont très convaincants puisque l'on a la possibilité de faire intervenir des contraintes très variées (inclinaison de la source lumineuse, alignement de plusieurs néons, etc.). Mais cette liberté est limitée par l'écriture obligatoire d'un script où la description des contraintes et des objectifs se fait à un très bas niveau. Ceux-ci sont saisis sous une forme très primitive, s'appuyant nécessairement sur des considérations géométriques ou physiques extrêmement détaillées.

Le projet CASSILDE a été initié par les auteurs de [CSF99a] et [CSF99b] pour offrir à l'utilisateur un moyen de spécifier ses désirs sans avoir à fabriquer un script complexe. Aucun article n'a été publié à ce jour en relation avec ce projet.

On notera pour finir qu'une approche en terme de modélisation déclarative ([LD95], [PLE95], [GAI03]) serait probablement assez fructueuse pour remédier aux inconvénients que nous venons d'exposer.

2.5.5 Conclusion.

La grande majorité des algorithmes recensés dans cet état de l'art ont recours à des techniques d'optimisation (Même l'utilisation d'un algorithme génétique est une sorte de technique d'optimisation [GOL94]). La formulation mathématique des problèmes amène à des équations qui ne sont pas résolubles formellement, soit de façon absolue, soit dans des temps raisonnables. Il est alors usuel de recourir à des procédés numériques d'approximation employés aussi en physique.

D'autre part, on cherche à remplir au mieux les exigences de l'utilisateur en matière d'éclairage de la scène. La plupart des problèmes d'éclairage inverse possèdent un grand nombre de solutions satisfaisantes, mais satisfaisantes uniquement d'un point de vue

théorique – et non pas pour le concepteur qui ne jugera pas les positions des sources lumineuses obtenues très réalistes. De plus, il est possible que la technique d'optimisation élimine des solutions que l'utilisateur pourrait trouver intéressantes mais auxquelles il ne pensait pas a priori. Sur ce point, une méthodologie similaire à celle de la modélisation déclarative ([LD95], [PLE95], [GAI03]) devrait apporter une aide au concepteur cherchant à obtenir un certain type d'éclairage pour une scène.

A propos de ces techniques d'optimisation, [PP01a] soulignent la diversité des approches. Il n'y a pas véritablement de schéma directeur, de modèle universellement étudié, et le choix des algorithmes de résolution n'est le plus souvent pas justifié.

Les résultats les plus prometteurs sont certainement ceux de [CSF99a] et [CSF99b]. La quantité physique analysée étant la radiance, il est possible de travailler avec un éclairage indirect et des surfaces ayant des propriétés diverses. La qualité et la variété des résultats (avec différentes formes de sources lumineuses, ainsi que différentes orientations pour celles-ci) tranchent avec celles des algorithmes se plaçant dans des modèles physiques plus limités. Le défaut de l'outil présenté est incontestablement sa difficulté de manipulation par un non-initié (puisque le script que l'on doit écrire afin de spécifier ses désirs en matière d'éclairage est très proche de l'algorithme de recuit simulé qui est mis en œuvre lors de la phase de résolution). Il serait plus agréable au concepteur de décrire de façon intuitive - en recourant par exemple à la notion d'ambiance - les objectifs d'éclairage qu'il souhaite voir remplis.

[KPC93] est le seul article où est pris en compte le concept d'ambiance. A l'aide de tests psychophysiques, [KPC93] prétendent ramener des notions comme la clarté d'une scène à des données numériques. Mais leurs résultats prêtent à discussion et le nombre de concepts introduits demeure assez limité. En outre, l'utilisateur doit introduire plusieurs données numériques (des coefficients servant à donner plus ou moins de poids à diverses fonctions objectifs). On ne peut donc dire que le logiciel présenté soit des plus simples à manier.

Ainsi, nous voyons que les méthodes de résolution proposées pour les problèmes d'éclairage inverse ne sont pas pleinement justifiées d'un point de vue théorique et mathématique. Quant à la formulation non quantitative de ces problèmes, beaucoup de travail reste à faire, puisque les auteurs se sont jusqu'ici essentiellement concentrés sur la phase de résolution.

3 Utilisation d'une méthode de Monte-Carlo pour le problème de l'éclairage inverse direct.

3.1 Introduction.

L'état de l'art précédent a montré l'importance du modèle d'illumination pour la formulation et la résolution d'un problème d'éclairage inverse. Pour nos travaux dans le cadre de l'éclairage inverse, nous avons choisi comme modèle d'illumination la radiosité. Ce modèle d'éclairage présente plusieurs avantages :

- Il est plus élaboré que celui de Phong qui est seulement une formule empirique.
- Il repose sur une base physique.
- Il est plus simple à manipuler qu'un modèle gérant la radiance en un point de la scène (et réalise ainsi un compromis entre ce genre de modèle et le modèle de Phong).
- Il permet une exploration de la scène sans autres calculs que ceux de visibilité dans un Z-buffer une fois que les radiosités des éléments ont été calculées.

Nous allons exposer une nouvelle méthode d'éclairage inverse basée sur une technique de Monte-Carlo. Avant de détailler cette méthode, nous commencerons par présenter

brièvement les principaux concepts rencontrés en radiosité ainsi que ceux attachés aux méthodes de Monte-Carlo.

Nous n'aborderons que dans les paragraphes 4 et 5 les problèmes liés à la gestion de l'aspect déclaratif des souhaits de l'utilisateur en matière d'éclairage.

3.2 La radiosité.

La radiosité est une méthode d'illumination globale, où l'on dérive sous certaines conditions des équations moins complexes que l'équation de rendu de Kajiya [KAJ86] (qui est une équation intégrale de Fredholm du second ordre) :

$$L(x,\omega) = L_e(x,\omega) + \int_S \rho(x,\omega,\omega')L(x',\omega')G(x,x')dA'$$

où :

- x et x' sont des points sur des surfaces de la scène.
- ω et ω' sont des directions.
- $L(x,\omega)$ est la radiance, s'exprimant en $Watt.m^{-2}.sr^{-1}$, au point x et dans la direction ω.
- $L_e(x,\omega)$ est la radiance émise.
- S désigne l'ensemble des surfaces de la scène.
- $\rho(x,\omega,\omega')$ est la fonction de réflectance bidirectionnelle : c'est le rapport entre la radiance réfléchie dans la direction ω et l'éclairement de la surface selon la direction ω'.
- G(x,x') est un terme de nature géométrique contenant notamment une fonction de visibilité V(x,x') :

$$G(x,x') = \frac{V(x,x')\cos\theta\cos\theta'}{r^2}$$

  - V(x,x') vaut 1 si les point x et x' ne sont pas masqués par d'autres éléments de la scène ; elle vaut 0 sinon.
  - θ (respectivement θ') est l'angle entre la direction ω (respectivement ω') et la normale à x (respectivement x'). r représente lui la distance d(x,x').
- dA' est un élément d'aire différentiel.

En supposant que le milieu est non-participant et que les surfaces sont toutes lambertiennes (c'est-à-dire parfaitement diffuses), dans un environnement clos l'équation précédente se ramène à l'équation suivante exprimée en termes de radiosités :

$$B(x) = E(x) + \rho(x)\int_S B(x')\,V(x,x')\frac{\cos\theta\cos\theta'}{\pi r^2}\,dA'$$

Cette équation se nomme *l'équation de radiosité*.

B(x) désigne la radiosité au point x, exprimée en $Watt.m^{-2}$.

E(x) est l'émittance en x ; de même unité que la radiosité, elle est uniquement non nulle pour les sources lumineuses.

ρ(x) est la réflectance diffuse en x. Il s'agit d'une quantité comprise entre 0 et 1.

L'équation de radiosité reste une équation intégrale extrêmement difficile à résoudre formellement. Pour approcher la fonction B, on discrétise la scène en carreaux dans lesquels la radiosité, l'émittance et la réflectance sont constantes. D'un point de vue mathématique, ceci revient à approcher la fonction B par une fonction étagée $\sum_k B_k \mathbf{I}_{P_k}$ où les P_k sont les carreaux de la scène et B_k la valeur de la radiosité pour le carreau P_k.

On se retrouve alors avec le système d'équations :

$$B_i = E_i + \rho_i \sum_{j=1}^{n} F_{ij} B_j$$

où n désigne le nombre de carreaux en lesquels on a découpé la scène.

F_{ij} est appelé facteur de forme entre les carreaux P_i et P_j. Cette quantité représente la quantité d'énergie quittant le carreau P_i et atteignant directement le carreau P_j. Le facteur de forme dépend uniquement de la géométrie de la scène.

En utilisant l'hémisphère Ω autour du carreau P_i, on a pour F_{ij} l'expression suivante :

$$F_{ij} = \frac{1}{\pi A_i} \int_{A_i} \int_{\Omega} V_{ij} \cos\theta d\omega \, dA_i$$

Avec les méthodes d'analyse numérique pour la résolution du système d'équations de la radiosité (comme Gauss-Seidel ou Southwell), c'est le calcul des facteurs de forme qui prend le plus de temps.

L'une des techniques d'évaluation des facteurs de formes met en œuvre un procédé de Monte-Carlo [SHI91]. C'est celle-ci que nous allons expliciter à présent.

3.3 Les méthodes de Monte-Carlo.

Les méthodes de Monte-Carlo sont des méthodes de nature probabiliste permettant de calculer de façon approchée une intégrale. Contrairement aux méthodes numériques de quadrature (comme la méthode des rectangles ou la méthode de Simpson), l'efficacité des méthodes de Monte-Carlo ne dépend pas de la dimension de l'intégrale que l'on essaie d'évaluer, ni des propriétés de continuité ou de dérivabilité de la fonction à intégrer.

Dans le cadre de la radiosité, les techniques de Monte-Carlo ont été utilisées pour calculer les facteurs de forme. Elles ont aussi été employées pour la résolution de l'équation de rendu de Kajiya [KAJ86], équation intégrale dont on trouve une solution approchée à l'aide d'une marche aléatoire.

3.3.1 Principe de la méthode de Monte-Carlo.

On cherche à évaluer l'intégrale :

$$I = \int_0^1 f(x)dx$$

Remarques :

- Pour faciliter l'exposition de la méthode, on se limitera au segment [0,1] comme domaine d'intégration, mais les résultats suivants restent valables pour des domaines différents.
- La fonction f appartient à $L^2([0,1],\mu)$ où μ est la mesure de Lebesgue. Ceci est nécessaire pour pouvoir considérer la variance des estimateurs que l'on va construire.

Soit X une variable aléatoire réelle dont la loi suit la loi uniforme sur le segment [0,1]. Considérons l'estimateur suivant :

$$\langle I_1 \rangle = f(X)$$

L'espérance vaut : $E(\langle I_1 \rangle) = I$, donc l'estimateur $\langle I_1 \rangle$ est sans biais.

D'autre part, la qualité de cet estimateur nous est fournie par sa variance :

$$V(\langle I_1 \rangle) = \int_0^1 f(x)^2 dx - I^2$$

L'estimateur $\langle I_1 \rangle$, appelé estimateur primaire, possède en général une variance assez élevée.

Pour diminuer la variance, la méthode de Monte-Carlo fait appel à plusieurs échantillons et calcule la moyenne des estimateurs primaires associés.

On introduit donc un estimateur secondaire $\langle I_2 \rangle$ défini par :

$$< I_2 >= \frac{1}{N} \sum_{i=1}^{N} f(X_i)$$

L'espérance de cet estimateur est :

$$E(<I_2>) = I$$

donc l'estimateur est sans biais.

Quant à sa variance, elle vaut :

$$V(< I_2 >) = \frac{V(< I_1 >)}{N}$$

L'écart type associé à l'estimateur $<I_2>$ nous indique donc qu'il y a une convergence vers l'intégrale I de l'ordre de $O(\sqrt{N})$.

3.3.2 Echantillonnage d'importance.

La méthode présentée dans le paragraphe précédent peut être améliorée afin de réduire encore la variance de l'estimateur intervenant pour l'approximation de l'intégrale.

Considérons une densité de probabilité p telle que $p(x) > 0$ chaque fois que $f(x) \neq 0$.

Avec une variable aléatoire X dont la densité de probabilité est p, on introduit alors l'estimateur primaire suivant :

$$< I_1 >= \frac{f(X)}{p(X)}$$

$<I_1>$ est un estimateur sans biais, car on a :

$$E(< I_1 >) = \int_0^1 \frac{f(x)}{p(x)} p(x) dx = I$$

Par ailleurs, on a :

$$V(< I_1 >) = E\left(\left(\frac{f(X)}{p(X)}\right)^2\right) - \left(E\left(\frac{f(X)}{p(X)}\right)\right)^2$$

$$= \int_0^1 \frac{f(x)^2}{p(x)} dx - I^2$$

En considérant à présent N variables aléatoires indépendantes de densité de probabilité p, on construit de façon similaire au paragraphe précédent un estimateur secondaire :

$$< I_2 >= \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)}$$

$<I_2>$ est sans biais ; quant à sa variance, elle vaut :

$$V(< I_2 >) = \frac{V(< I_1 >)}{N}$$

$$= \frac{1}{N}\left(\int_0^1 \frac{f(x)^2}{p(x)} dx - I^2\right)$$

Pour réduire la variance de l'estimateur $<I_1>$, il convient de choisir avec attention la fonction de densité de probabilité p.

L'échantillonnage d'importance consiste à choisir p de telle sorte qu'elle ait une forme similaire à f. D'un point de vue théorique, le meilleur choix possible est :

$$p(x) = \frac{|f(x)|}{I},$$

mais il est bien évident que l'on ne connaît pas la valeur I (puisque c'est elle que nous cherchons à calculer).

Il faut aussi noter qu'il n'est pas toujours facile d'échantillonner une variable aléatoire ayant une densité de probabilité quelconque. Si l'on peut calculer explicitement la fonction de répartition P associée à p, puis sa fonction inverse, alors $X = P^{-1}(U)$ où U suit la loi uniforme sur [0,1] aura p comme densité de probabilité. Bien que d'autres techniques d'échantillonnage (méthode du rejet) soient applicables, dans un échantillonnage d'importance, on choisit généralement p de telle sorte que l'on puisse mettre en œuvre le procédé d'échantillonnage venant d'être décrit.

3.3.3 Application au calcul du facteur de forme.

Le facteur de forme entre deux carreaux P_i et P_j peut s'exprimer sous la forme :

$F_{ij} = \frac{1}{\pi A_i} \int_{A_i} \int_{\Omega} V_{ij}(\theta, \varphi, x) \cos\theta \sin\theta d\theta d\varphi dA_i$ où :

- Ω désigne l'hémisphère au-dessus de P_i,
- (θ,φ) une direction (associée à un angle solide différentiel $d\omega$ sur l'hémisphère),
- V_{ij} la fonction de visibilité valant 1 si le carreau P_j est visible depuis le carreau différentiel dA_i de centre x et dans la direction (θ,φ), et 0 sinon.

On effectue un échantillonnage d'importance en prenant pour densité de probabilité :

$$p(\theta, \varphi, x) = \frac{\cos\theta \sin\theta}{\pi A_i}$$

On détermine alors N échantillons à partir des formules :

$$\Theta = Arc\cos\sqrt{U_1}$$

$$\Phi = 2\pi U_2$$

(où U_1 et U_2 sont des variables aléatoires suivant une loi uniforme sur [0,1]) et X qui est une variable aléatoire uniforme sur A_i.

L'estimateur $<F_{ij}>$ s'écrit alors :

$$< F_{ij} >= \frac{1}{N} \sum_{k=1}^{N} V_{ij}(\Theta_k, \Phi_k, X_k)$$

On a ainsi une interprétation élémentaire de l'estimateur obtenu : on approche le facteur de forme F_{ij} par le nombre de rayons lancés depuis le carreau P_i et atteignant le carreau P_j divisé par le nombre total N de rayons lancés (Les caractéristiques géométriques des rayons – à savoir : origine et direction – sont bien sûr fournies par les échantillons de X, Θ et Φ).

3.4 La méthode d'éclairage inverse.

3.4.1 Introduction.

L'algorithme d'éclairage inverse que nous proposons se place dans le cadre de la radiosité et se limite dans un premier temps à un éclairage direct (On peut retrouver une présentation de cet algorithme dans [JPP02a]). Recourant à la technique de Monte-Carlo à base de lancers de rayons exposée précédemment, il prolonge les travaux de [OH95] qui ont été effectués dans le domaine des échanges d'énergie calorique et n'ont été testés que pour des environnements en deux dimensions.

3.4.2 Une nouvelle méthode d'éclairage inverse.

3.4.2.1 La problématique.

A partir d'un ensemble O de carreaux à éclairer, on cherche à placer une source lumineuse, qui sera un nouveau carreau ajouté à la scène, dont la forme est un quadrilatère et qui doit satisfaire à certaines conditions d'éclairage exprimées sous forme d'un intervalle de valeurs pour le flux lumineux de chacun des carreaux de O.

Les conditions d'éclairage se présentent sous forme d'un intervalle car elles sont exprimées initialement de façon déclarative. Nous avons vu dans la partie 2 que nous recourions à une représentation sous forme de sous-ensembles flous des propriétés demandées par l'utilisateur. Un intervalle est ensuite calculé à partir du sous-ensemble flou grâce à une α-coupe. C'est donc cet intervalle qui sert de base à l'algorithme d'éclairage inverse que nous allons maintenant présenter.

Remarquons de plus que contrairement aux autres méthodes d'éclairage inverse en radiosité, la source de lumière n'est pas l'un des carreaux du maillage initial de la scène. Ceci va nécessiter un remaillage de la surface sur laquelle on ajoute le carreau

3.4.2.2 La recherche des carreaux pouvant jouer le rôle de source lumineuse.

On utilise pour cela une méthode de Monte-Carlo élémentaire. On va lancer des rayons aléatoirement depuis chaque carreau à éclairer et noter les carreaux de la scène que ces rayons couperont en premier.

On s'intéresse tout d'abord à l'éclairage d'un seul carreau.

On recueille dans une liste L les carreaux capables d'éclairer directement le carreau à illuminer appartenant à la liste O. Quand on n'ajoute plus de nouveau carreau à la liste L au bout de MaxRayons lancés, on arrête de lancer des rayons. On considère que l'on a obtenu dans L tous les carreaux directement visibles depuis le carreau à éclairer.

On peut ensuite évaluer le facteur de forme entre le carreau à illuminer et les carreaux contenus dans la liste L. Ceci nous aidera pour la détermination de l'émittance à associer au carreau qui servira de source de lumière.

L'algorithme suggéré est le suivant :

Entrée :

- P_i carreau de la scène à éclairer.

Sortie :

- L liste des carreaux directement visibles depuis P_i
- $\{F_{ij}$ / l'indice j permettant de décrire l'ensemble des carreaux de L$\}$

Procédure Rechercher_Sources_Lumineuses_Pour_Carreau

```
// Détermination des carreaux vus directement depuis Pi
L ← ∅
nombre_Rayons ← 0
répéter
    nouveau_Carreau ← faux
    On lance un rayon aléatoirement depuis le carreau Pi
    Soit Pj le carreau atteint en premier par le rayon
    si Pj ∉ L alors
        L ← L ∪ Pj
        nouveau_Carreau ← vrai
```

```
        finsi
        Incrémenter(nombre_Rayons_Reçus(Pj))
        Incrémenter(nombre_Rayons)
        si nouveau_Carreau = faux alors
                Incrémenter(déjà_Atteint)
        sinon
                déjà_Atteint ← 0
        finsi
    jusqu'à déjà_Atteint = MaxRayons

    // Calcul de l'approximation du facteur de forme Fij
    pour tout carreau Pj ∈ L faire
        FacteurForme(i,j) ← nombre_Rayons_Reçus(Pj) / nombre_Rayons
    finpourtout

finProcédure
```

Remarque sur les résultats obtenus :

A partir d'un certain seuil pour MaxRayons, la liste L augmente très peu. Les carreaux qui s'ajoutent alors à L ont peu d'influence sur le carreau à éclairer. Ils présentent pour l'instant peu d'intérêt :

- parce qu'ils peuvent mener à une contradiction en éclairant par ailleurs une zone que l'utilisateur voudrait laisser dans l'ombre.
- parce que s'ils ont une véritable importance pour les objectifs à atteindre, ils seront pris en compte par les carreaux à éclairer à proximité de celui que l'on vient d'étudier (C'est en effet un cas de figure assez courant : On s'intéresse généralement à l'éclairage d'un ensemble de carreaux formant un objet ou une partie d'un objet de la scène, et non pas à l'éclairage d'un carreau isolé).

Afin de limiter les temps de calcul, il convient donc de trouver une bonne valeur pour MaxRayons.

Mais déterminer une heuristique de bonne qualité pour MaxRayons est difficile, puisque cette valeur dépend fortement de la géométrie de la scène.

C'est pourquoi nous avons développé une variante de l'algorithme précédent dans laquelle MaxRayons n'intervient plus.

A chaque étape, on ne lance qu'un nombre restreint de rayons (de l'ordre de 100). Si à la fin d'une étape, le nombre de nouveaux carreaux atteints par les rayons est supérieur à 20% du nombre total de rayons lancés, on réitère le processus.

En pratique, on constate que l'on obtient, dans les mêmes temps de calcul, la quasi totalité des carreaux que l'on a en ajustant empiriquement la valeur de MaxRayons pour l'algorithme précédent.

Etudions à présent l'éclairage d'un ensemble de carreaux.

On peut prendre la réunion des listes obtenues pour les différents appels à la procédure Rechercher_Sources_Lumineuses_Pour_Carreau.

Pour ce qui suit, appelons cette liste Λ.

Remarque :

Si l'on veut éclairer une facette constituée de plusieurs carreaux, il n'est pas intéressant de rechercher les sources de lumières possibles pour tous ces carreaux. Avec un petit nombre de carreaux sélectionnés sur la facette, on retrouve la totalité (ou la quasi-totalité) des carreaux pouvant jouer le rôle de sources d'éclairage direct. On gagne ainsi considérablement en temps de calcul.

Il faut donc choisir les « bons » carreaux d'une facette (ou d'un objet) à éclairer.

Nous pouvons envisager les idées suivantes :

Pour chaque sommet de la facette, on sélectionne le (ou l'un des) carreau(x) comprenant ce sommet. On rajoute de plus dans la liste des « bons » carreaux le carreau contenant l'isobarycentre de la facette.

Si l'on veut éclairer un objet (ou une partie d'un objet), on va être conduit à appliquer le procédé précédent à toutes les facettes de l'objet. La valeur de l'angle formé par deux facettes va autoriser des optimisations supplémentaires (gains obtenus grâce à moins de carreaux sélectionnés comme objectifs d'éclairage). Si l'angle géométrique entre deux faces est suffisamment grand, il sera possible de se dispenser d'avoir certains carreaux contigus dans la liste des carreaux de la scène à éclairer. La figure 43 illustre le procédé envisagé.

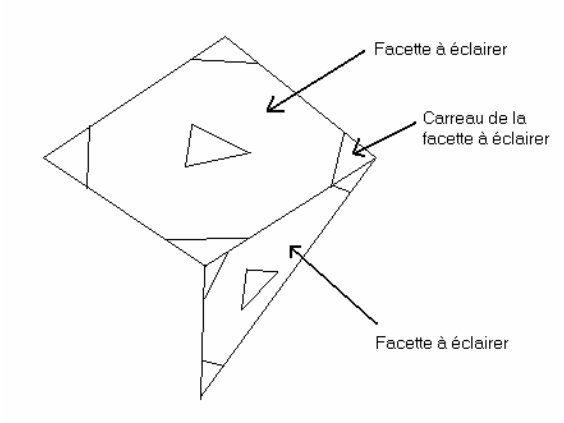


Figure 43. Choix des carreaux représentatifs de la surface à éclairer.

3.4.2.3 Suppression dans la liste Λ des carreaux ayant trop peu d'influence sur les carreaux à éclairer.

Quand on traite des scènes avec peu d'objets, il n'est pas rare que pour 5 carreaux à éclairer, on obtienne une liste Λ de beaucoup plus de 300 carreaux pouvant être des sources de lumière. De nombreux carreaux constituant les faces de la boîte englobante de la scène (boîte englobante nécessaire pour assurer l'hypothèse d'un environnement clos) sont en effet ajoutés à Λ s'il n'y a pas d'objet placé entre ces faces et le carreau à éclairer. Même si l'utilisateur fixe a priori des contraintes sur la position des sources d'éclairement potentielles (par exemple en imposant aux sources lumineuses d'être situées uniquement au plafond d'une pièce), trop de carreaux peu importants seront intégrés à Λ.

On décide donc d'éliminer un carreau j de la liste Λ si le facteur de forme F_{ij} est inférieur à une valeur de seuil (SeuilMin(i)) pour au moins l'un des carreaux i à éclairer.

Comme heuristique pour SeuilMin(i), nous proposons :

$$SeuilMin(i) = \frac{1}{Nombre\ de\ carreaux\ visibles\ depuis\ i}$$

i étant l'un des carreaux à illuminer, c'est-à-dire un élément de la liste O.

Grâce à cette heuristique, on diminue considérablement le nombre d'élément de Λ.

Un exemple de suppression obtenue :

Sur la figure 44, on peut voir une image où apparaissent en blanc toutes les sources lumineuses potentielles. Les carreaux que l'on souhaite éclairer sont placés sur le haut de la boîte verte à gauche de la scène.

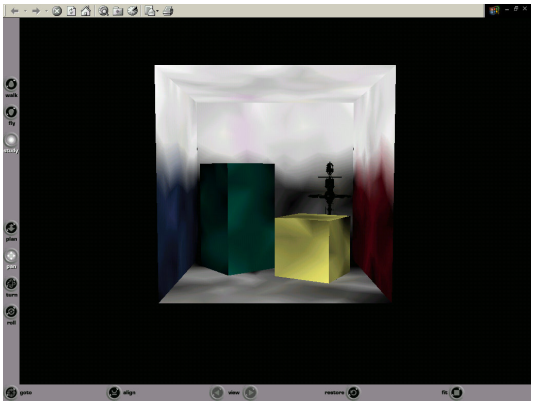


Figure 44. La totalité des sources lumineuses potentielles pour l'éclairement du haut de la boîte de gauche.

Sur la figure 45, on peut voir les sources lumineuses qui restent quand on supprime celles ayant un facteur de forme inférieur à une valeur de seuil, et que l'on se limite au plafond pour la position des sources de lumière potentielles :

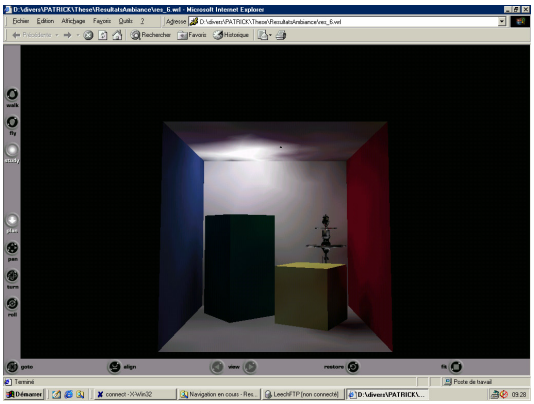


Figure 45. Les sources lumineuses intéressantes (et limitées au plafond) pour l'éclairement du haut de la boîte de gauche.

On peut considérer que l'on obtient une zone de placement pour les sources d'éclairages beaucoup plus réaliste de la sorte.

Remarque : Les carreaux qui apparaissent en blanc sont les sources de lumière potentielles auxquelles nous avons attribué une émittance très élevée pour produire cette image.

3.4.2.4 Ajout à la scène d'un carreau servant de source de lumière.

Une fois que la liste Λ, liste contenant les carreaux ayant un rôle important dans l'éclairage des carreaux de la liste O, a été déterminée comme nous l'avons vu au paragraphe précédent, il reste à préciser les caractéristiques d'un (ou plusieurs) carreau(x) éclairant(s).

Pour l'instant, on se cantonnera à un seul carreau émetteur (que l'on notera P_l), dont la forme est celle d'un quadrilatère.

Afin de placer le carreau émetteur, on construit un polygone convexe à partir des carreaux de la liste Λ. Pour cela, on utilise l'algorithme de Graham [GRA72] que l'on applique aux sommets des carreaux de la liste Λ.

On balaie ensuite le polygone convexe afin de trouver les positions acceptables pour le carreau émetteur.

Une fois que la position du carreau émetteur a été fixée, il reste à remailler la face dans laquelle on l'introduit. Pour cela, on commence par déterminer les carreaux affectés par l'introduction de P_l dans la scène.

Pour chacun des carreaux triangulaires dont l'intersection avec P_l est non vide, 5 cas principaux se présentent, cas correspondant aux nombres de sommets de P_l situés à l'intérieur du triangle. Ces cas se divisent ensuite éventuellement en sous cas selon le nombre de sommets du triangle situés à l'intérieur du carreau P_l.

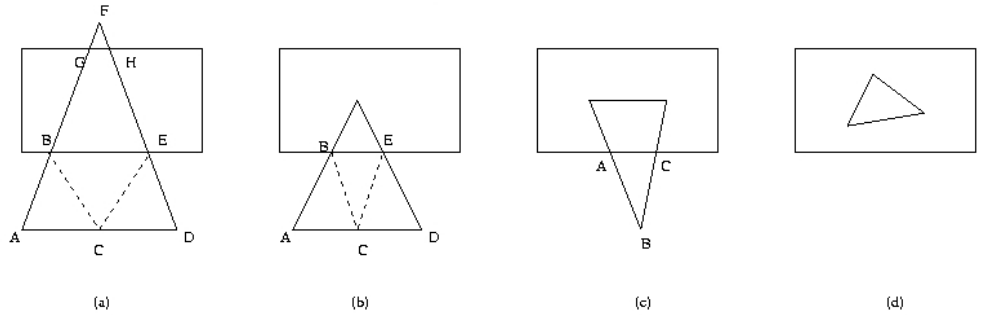


Figure 46. Cas d'un triangle sans sommets de P_l dans son intérieur.

Dans les 4 cas de la figure 46, on supprime le carreau triangulaire original.

Sur les schémas de la figure 46, le nombre de sommets du triangle à l'intérieur de P_l varie de 0 à 3.

Dans la configuration (a), on introduit comme carreaux supplémentaires dans la scène les 4 triangles (ABC), (BCE), (CDE) et (FGH). Le point C est choisi sur le segment [AD].

Dans la configuration (b), on ajoute à la scène les 3 triangles (ABC), (BCE) et (CDE). Le point C est choisi sur le segment [AD] de façon similaire au cas (a).

Dans la configuration (c), on ajoute le seul triangle (ABC).

Enfin, dans la configuration (d), on ne rajoute aucun triangle après avoir retiré le carreau initial.

Dans les 2 cas de la figure 47, on supprime le carreau triangulaire original.

Sur les schémas de la figure 47, le nombre de sommets du triangle à l'intérieur de P_l varie de 0 à 1.

Dans la configuration (a), on introduit comme carreaux supplémentaires dans la scène les 4 triangles (ABE), (ABD), (BCD) et (CDF).

Dans la configuration (b), on ajoute à la scène les 3 triangles (ACD), (ABC) et (BCE).

Dans les 2 cas de la figure 48, on supprime le carreau triangulaire original.

Sur les schémas de la figure 48, le nombre de sommets du triangle à l'intérieur de P_l varie de 0 à 1.

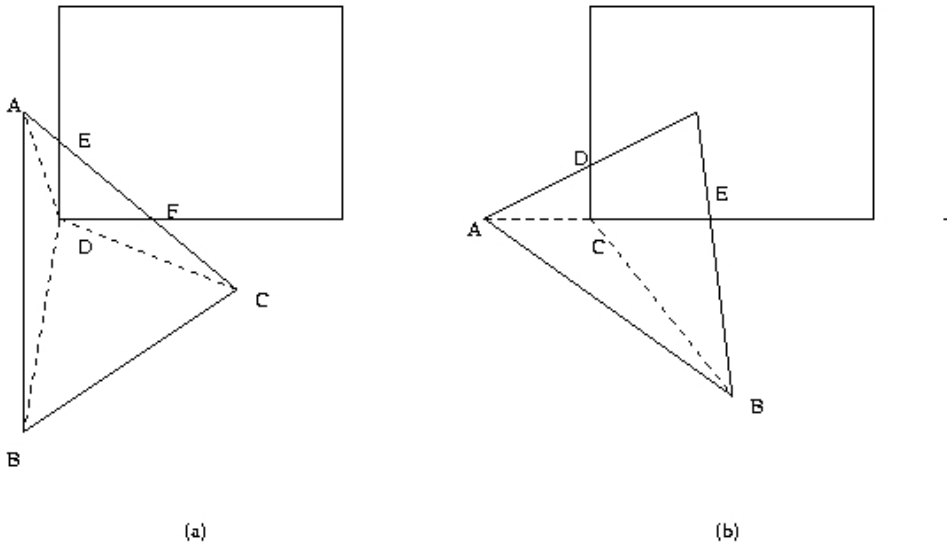


Figure 47. Cas d'un triangle avec 1 sommet de P_I dans son intérieur.

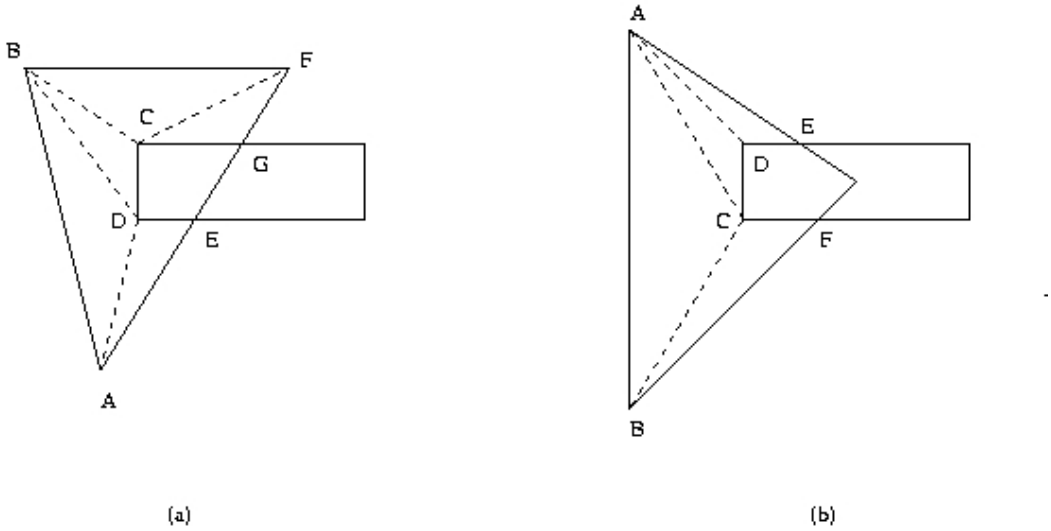


Figure 48. Cas d'un triangle avec 2 sommets de P_I dans son intérieur.

Dans la configuration (a) de la figure 48, on introduit comme carreaux supplémentaires dans la scène les 5 triangles (BCF), (BCD), (ABD), (ADE) et (CFG).

Dans la configuration (b), on ajoute à la scène les 4 triangles (ADE), (ACD), (ABC) et (BCF).

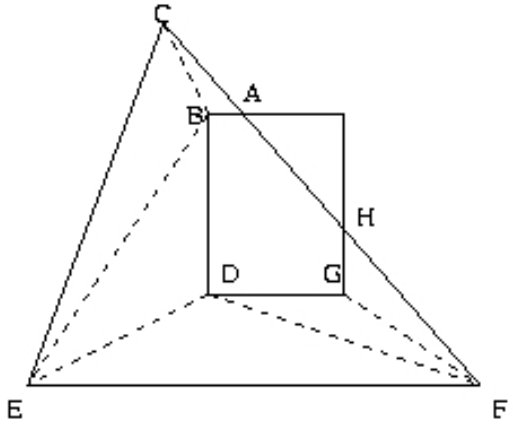


Figure 49. Cas d'un triangle avec 3 sommets de P_I dans son intérieur.

Dans le cas de la figure 49, on supprime le carreau triangulaire initial et on ajoute 5 nouveaux triangles à la scène : (ABC), (CBE), (BDE), (DEF), (DFG), (FGH).

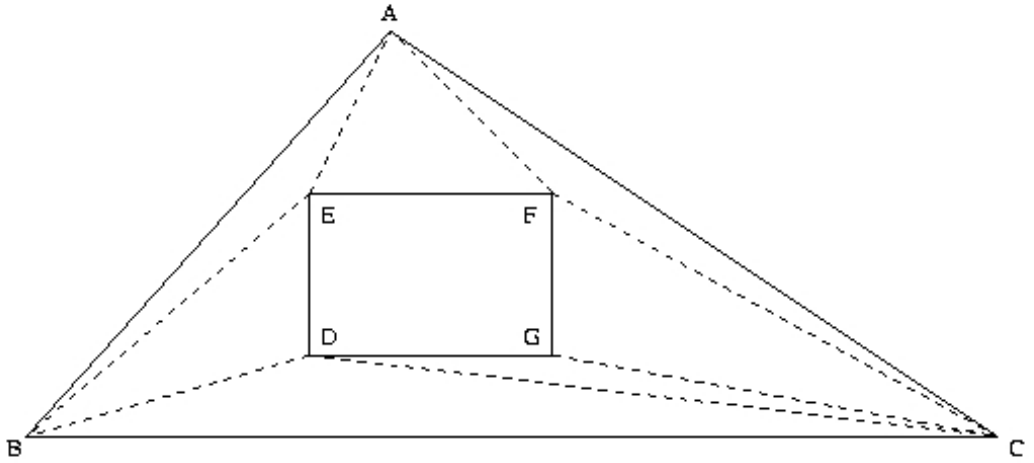


Figure 50. Cas d'un triangle avec 4 sommets de P_I dans son intérieur.

Dans la configuration de la figure 50, on supprime le triangle initial (ABC) et on rajoute (en plus du carreau rectangulaire servant de source lumineuse) les 7 triangles : (ABE), (BDE), (BCD), (CDG), (CFG), (ACF) et (AEF).

Quant aux carreaux rectangulaires qui pourraient être affectés par le remaillage, on se contente de les diviser en deux carreaux triangulaires selon l'une de leurs diagonales (ce qui permet de se ramener à l'étude précédente).

Remarque : Puisque l'on se limite dans cette partie à un éclairage direct, la qualité du maillage de la face où se trouve positionné le carreau éclairant n'a pas d'importance. Par contre, on pourrait effectuer un maillage de discontinuité [LTG92] du reste de la scène.

Quand on a obtenu une position convenable et remaillé la scène en conséquence, on essaie d'attribuer une émittance au carreau éclairant P_l en tenant compte des valeurs autorisées pour le flux lumineux des carreaux à éclairer.

Examinons la relation qui existe entre la puissance lumineuse ϕ_i d'un carreau P_i à illuminer et la radiance émise L^e de la source lumineuse P_l. Comme on se restreint à un éclairage direct, on a l'égalité :

$$\Phi_i = \pi\rho_i A_i F_{il} L^e$$

où A_i désigne l'aire du carreau P_i.

En utilisant la relation de réciprocité (à savoir : $F_{ji}A_j = F_{ij}A_i$ pour des carreaux P_i et P_j), on aboutit à :

$$L^e = \frac{\Phi_i}{\pi\rho_i F_{li} A_l}$$

Pour chaque carreau P_i, on a un encadrement du type :

$$\min_i \leq \Phi_i \leq \max_i$$

qui se transforme ainsi en :

$$\frac{\min_i}{\pi\rho_i F_{li} A_l} \leq L^e \leq \frac{\max_i}{\pi\rho_i F_{li} A_l}$$

On s'aperçoit donc que l'on doit calculer les facteurs de forme F_{li} entre le carreau émetteur P_l et les carreaux à éclairer. On peut employer de nouveau la méthode de Monte-Carlo qui nous avait permis de déterminer la liste L.

En prenant en compte l'ensemble O des carreaux à illuminer, on peut de cette façon calculer les valeurs acceptables pour L^e, ou observer que l'on n'a pas de solution possible pour cette position du carreau éclairant. Avec une discrétisation de l'intervalle des valeurs convenables pour L^e, on parvient alors à fournir plusieurs solutions à l'utilisateur, pour la même position du carreau émetteur.

Les figures 51 et 53 présentent deux exemples de scènes obtenues avec des positions différentes pour la source lumineuse. Les figures 52 et 54 montrent l'éclairage obtenu sur les carreaux que l'on veut illuminer.

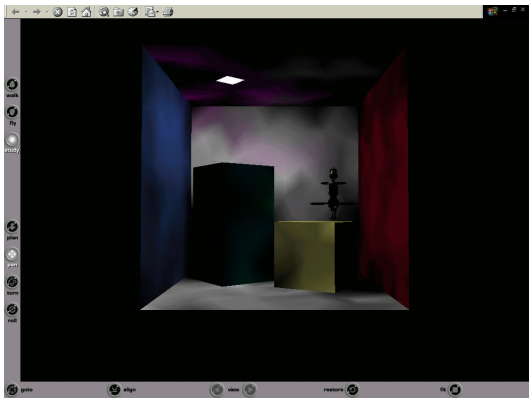


Figure 51. Première solution obtenue.

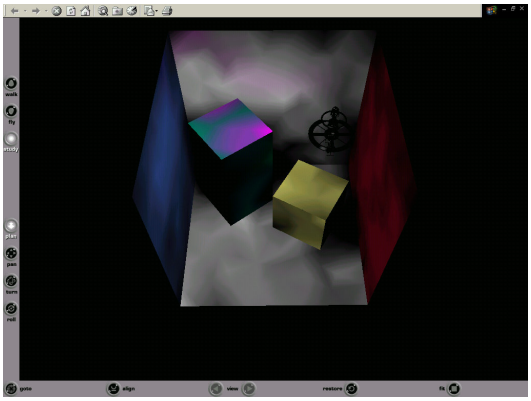


Figure 52. Les carreaux à éclairer sont situés sur le haut de la boîte de gauche. Vue de ces carreaux avec la première solution.

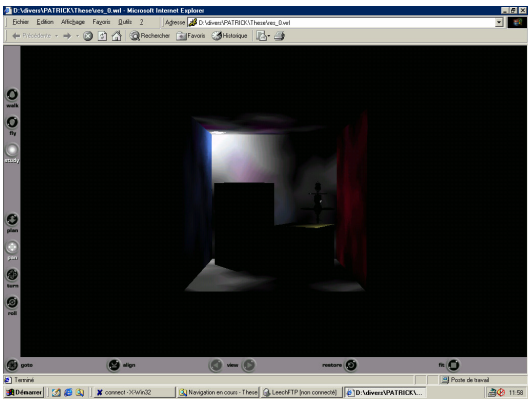


Figure 53. Dernière solution obtenue.

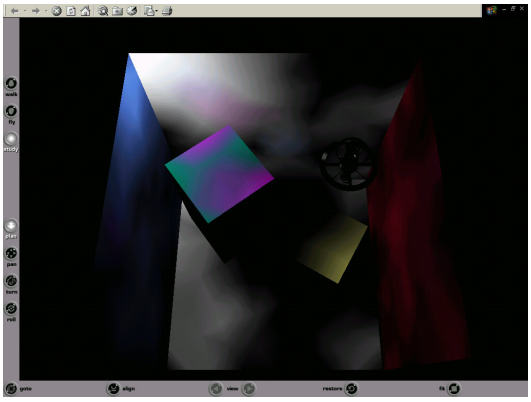


Figure 54. L'éclairage des carreaux sur le haut de la boîte de gauche. Vue de ces carreaux avec la dernière solution calculée.

3.4.2.5 Quelques temps de calcul pour la scène servant d'illustration à la méthode.

Nous allons donner maintenant des temps de calcul correspondant aux figures 51 à 54.

La scène étudiée est constituée de 5938 carreaux. Les carreaux à éclairer sont situés sur le haut de la boîte de gauche. Dans les spécifications d'éclairage, on a comme contrainte le fait que la source de lumière doit être placée au plafond (Ce dernier est constitué initialement de 154 carreaux).

Les résultats que nous présentons ont été obtenus avec la configuration suivante :

- Bi-processeur Pentium III Xeon à 1,2 Ghz.
- 768 Mo de RAM.
- Système d'exploitation FreeBSD 4.9.

Nous fournissons ci-dessous un listing des résultats engendrés par notre implémentation de notre algorithme d'éclairage inverse.

```
----> Construction de la scène.
Statistiques :
      Nombre de sommets                  : 6455
      Nombre de carreaux                 : 5938
      Temps                              : < 1s

----> Chargement des objectifs.
Statistiques :
      Nombre de carreaux à éclairer                        : 6
      Nombre de sources potentielles (avec contrainte)     : 154
      Temps                                                : < 1s


############################################################
Résolution du problème d'illumination inverse.
Lancement de la méthode de résolution de Monte-Carlo de base.


Recherche des carreaux pouvant faire office de sources lumineuses.
Nombre de sources lumineuses potentielles pour le carreau 150 : 329
Nombre de rayons lancés : 8203
Nombre de sources lumineuses potentielles pour le carreau 151 : 327
Nombre de rayons lancés : 7957
Nombre de sources lumineuses potentielles pour le carreau 152 : 330
Nombre de rayons lancés : 8216
Nombre de sources lumineuses potentielles pour le carreau 153 : 328
Nombre de rayons lancés : 8210
Nombre de sources lumineuses potentielles pour le carreau 154 : 320
Nombre de rayons lancés : 11555
Nombre de sources lumineuses potentielles pour le carreau 155 : 327
Nombre de rayons lancés : 11302
Sources lumineuses potentielles déterminées.
Nombre de sources lumineuses potentielles : 332

Prise en charge d'une contrainte sur la position de la source lumineuse.
Nombre de sources lumineuses potentielles : 153

Temps de calcul pour la détermination des sources potentielles : 2s

Recherche des sources lumineuses importantes.
Nombre de carreaux à retirer de la liste des sources lumineuses
potentielles : 119
Sources lumineuses importantes déterminées.
Nombre de sources lumineuses importantes  : 34
Temps de calcul                           : < 1s

Recherche de la position et de l'émittance des carreaux pouvant être des
sources lumineuses.
Nombre de points dans l'adhérence du polygone : 30
Nombre de points sur l'enveloppe convexe du polygone : 10
```

```
Nombre de sommets après élimination des points alignés : 6

Solution trouvée.
Radiance émise trouvée : (318.31,318.31,318.31)
Lancement de la méthode de Monte-Carlo (Feda-Purgathofer)
Fichier VRML produit : ETAPES/res_sol1_1.wrl
Etape numéro 1
Temps de calcul  : 1s
Nombre de rayons : 1051

Solution trouvée.
Radiance émise trouvée : (318.31,318.31,318.31)
Lancement de la méthode de Monte-Carlo (Feda-Purgathofer)
Fichier VRML produit : ETAPES/res_sol2_1.wrl
Etape numéro 1
Temps de calcul  : < 1s
Nombre de rayons : 1048
```

(…)

```
Solution trouvée.
Radiance émise trouvée : (636.62,636.62,636.62)
Lancement de la méthode de Monte Carlo (Feda-Purgathofer)
Fichier VRML produit : ETAPES/res_sol58_1.wrl
Etape numéro 1
Temps de calcul  : < 1s
Nombre de rayons : 1179


Nombre de solutions trouvées : 58
Carreaux éclairants déterminés.

Temps de calcul pour la génération des scènes solutions : 20s

Temps de calcul total : 24s

*** FIN ***
```

Commentaires :

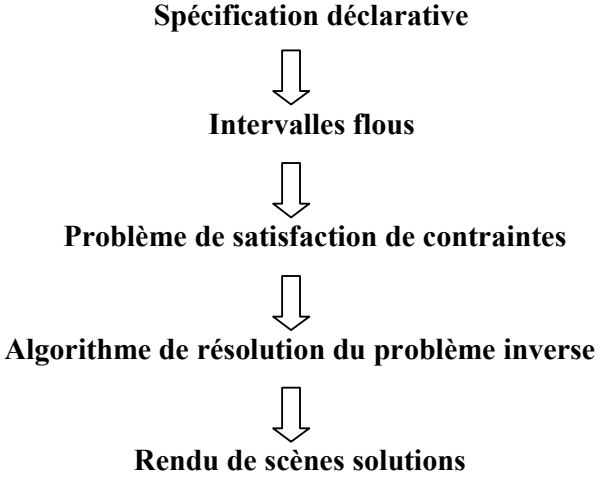
- Nos algorithmes ont été greffés sur le logiciel Heufora (*Heuristics for Radiosity*) de Vincent Jolivet.
- L'émittance de la dernière solution est plus élevée en raison de la position excentrée de la source lumineuse dans ce cas-là (cf. figure 53).
- La méthode de radiosité employée est celle de Feda-Purgathofer. On s'est limité à une seule itération de cette méthode de Monte-Carlo puisque l'on est dans le cas d'un éclairage direct.
- Les scènes produites sont stockées dans des fichiers VRML.

4 Gestion de l'aspect déclaratif des spécifications d'éclairage.

4.1 Introduction.

Dans le paragraphe 3, nous avons présenté une méthode d'éclairage inverse sans pleinement tenir compte de notre cadre de travail. Nous allons voir quels ajustements sont induits par le fait que les objectifs d'éclairage sont spécifiés de façon déclarative par l'utilisateur.

Pour l'outil permettant une approche déclarative de l'éclairage, nous proposons le découpage en unités fonctionnelles suivant :



Ceci constitue un schéma de fonctionnement plus idéal que réaliste. Dans la pratique, l'algorithme traitant le problème d'éclairage inverse intervient juste après la spécification déclarative, en s'appuyant notamment sur certaines contraintes exprimées au niveau de la description qualitative de l'éclairage souhaité.

Il est possible de mettre en rapport les différents constituants du schéma précédent avec les trois principales étapes de la modélisation déclarative ([PLE95], [LD95], [GAI03]).

La phase de description de la modélisation déclarative correspond de façon évidente à la spécification déclarative du problème d'éclairage inverse.

La phase de génération est ici mise en œuvre par la résolution du problème de satisfaction de contraintes et l'algorithme résolvant le problème d'éclairage inverse. Le modèle interne est en grande partie constitué par des intervalles flous qui représentent les propriétés demandées par le concepteur.

La phase de prise de connaissance se limite pour l'instant au rendu de scènes solutions.

Dans les sections suivantes, nous allons expliquer comment l'algorithme d'éclairage inverse que nous avons explicité au 3.4 doit être ajusté pour s'insérer dans le schéma de fonctionnement que nous venons de présenter. Après un survol des méthodes de résolution des CSP, nous verrons quelles difficultés se posent avec les CSP que nous obtenons avec notre algorithme d'éclairage inverse (même une fois celui-ci modifié). Ceci nous conduira à redéfinir notre approche d'une méthode permettant de spécifier de façon déclarative des conditions d'éclairage d'une scène.

4.2 Modification de l'algorithme d'éclairage inverse pour l'adapter à la gestion de spécifications déclaratives.

La façon de procéder que nous avons présentée au 3.4.2 complique en fait fortement le travail de l'algorithme résolvant le problème de satisfaction de contraintes. En effet, la contrainte portant sur la radiance émise d'une source de lumière doit être vérifiée en dernier (après celles conditionnant le choix de la position, de la taille, etc.). C'est une contrainte d'une grande importance, et le fait que l'on doive l'examiner après avoir assigné une valeur aux autres inconnues du problème empêche de réaliser des optimisations pour l'exploration des intervalles de ces variables. Or la réduction d'intervalles (qui peut mener jusqu'à une contradiction) est au cœur des algorithmes traitant les problèmes de satisfaction de contraintes. Bien que l'on puisse toujours formuler notre problème à l'aide de contraintes, les

techniques habituelles de manipulation de celles-ci se révèleront toutefois complètement inefficaces. Ceci nous a donc conduit à repenser le fonctionnement de notre méthode d'éclairage inverse. Nous allons à présent développer une version mieux adaptée aux exigences des techniques de satisfaction de contraintes.

Les éléments de la scène pouvant contenir des sources d'éclairage vont préalablement être remaillés. Il n'est pas forcément obligatoire de déterminer un nouveau maillage pour toutes les parties de la scène puisque l'on peut avoir d'entrée des contraintes sur la position des sources de lumière (On peut les restreindre à être situées au plafond par exemple). Le nouveau maillage que l'on calcule se fait à base de carrés d'une taille suffisamment petite. De façon identique à l'algorithme présenté précédemment, on a ensuite plusieurs étapes de lancers de rayons à partir des carreaux à éclairer. A la fin du traitement d'un carreau à éclairer i, on a recueilli dans une liste L_i, non seulement les numéros des carreaux touchés par les rayons partant de i, mais aussi un intervalle pour la radiance émise à affecter au carreau atteint j, intervalle correspondant à la contrainte sur la puissance Φ_i voulue pour le carreau i et dont les bornes sont calculées à l'aide d'une estimation du facteur de forme F_{ij}. L_i se présente ainsi sous la forme :

$$L_i = \{(j, [L^i_{min}, L^i_{max}]) \,/\, j \text{ indice d'un carreau atteint}\}$$

Pour chaque carreau i à illuminer, on a une liste L_i de ce genre. La liste L prenant en compte tous les carreaux à éclairer est définie par :

$$L = \bigcap_{i \in I} L_i$$

$$:= \left\{ (j, \bigcap_{i \in I} [L^i_{min}, L^i_{max}]) \; tels\ que\ \forall i \in I, j \in L_i \; et \bigcap_{i \in I} [L^i_{min}, L^i_{max}] \neq \varnothing \right\}$$

où I désigne l'ensemble des carreaux à éclairer.

S'il y a incompatibilité dans les objectifs voulus par l'utilisateur, on obtiendra :

$$L = \varnothing$$

Ce sera par exemple le cas pour deux objets proches pour lesquels on demanderait des éclairages très différents. L'intersection des intervalles contenus dans les listes L_i sera alors vide, ce qui indiquera une contradiction dans les spécifications.

On peut ainsi assigner à chacun des carreaux de la liste L une radiance émise de telle sorte que les souhaits en matière d'éclairage du concepteur soient satisfaits.

Lorsque l'on introduira une source lumineuse dans la scène, source qui sera constituée par un ensemble de n carreaux de L, on divisera par n chacune des bornes L^i_{min} et L^i_{max} de ces carreaux afin de parvenir à l'éclairage voulu (On pourrait utiliser une autre combinaison linéaire dont la somme des coefficients vaudrait 1, mais cela donnerait une apparence non uniforme à la source lumineuse). S'il y a plusieurs sources de lumière, on devra diviser ces bornes par le nombre total de carreaux de L formant les sources d'éclairage.

4.3 Un rapide survol des méthodes de résolution des CSP.

4.3.1 Introduction aux CSP finis.

Nous allons présenter maintenant les principales méthodes employées pour résoudre les CSP (*Constraint Satisfaction Problems*). Pour un exposé plus détaillé des différents algorithmes et

techniques évoqués dans ce qui suit, on pourra se référer par exemple à [MS01], [RUT98], [TSA93] et [KUM92].

Formellement, un problème de satisfaction de contraintes est défini par un triplet (V,D,C) où :

- $V = \{X_1, X_2, \cdots, X_n\}$ est un ensemble de n variables.
- $D = \{D_1, D_2, \cdots, D_n\}$ est un ensemble de n domaines *finis* associés aux variables de V.
- C est l'ensemble des contraintes sur les variables. Si une contrainte fait intervenir k variables, alors elle est définie comme un sous-ensemble du produit cartésien des k domaines de ces variables.

Le fait de limiter les domaines à des ensembles finis amène à qualifier ces CSP de CSP finis.

Une solution du CSP consiste en un assignement pour chaque variable (c'est-à-dire une affectation d'une valeur de son domaine à la variable) de telle sorte que toutes les contraintes de C soient satisfaites.

Bien qu'une contrainte soit définie en extension, c'est-à-dire comme une relation sur un produit cartésien de domaines, il est possible qu'une contrainte soit donnée en intension, sous la forme d'une équation ou d'une inéquation par exemple.

Quand une contrainte ne fait intervenir que n variables, on parle de contrainte n-aire. Un CSP binaire est un CSP constitué uniquement de contraintes binaires ou unaires.

On démontre que l'on peut ramener un CSP n-aire à un CSP binaire équivalent. Ce résultat explique que la plupart des résultats sur les CSP soient présentés pour des CSP binaires (ce qui facilite la formulation des algorithmes et les preuves de théorèmes).

On représente habituellement les CSP binaires par des graphes où les nœuds correspondent aux variables et les arêtes à une contrainte entre deux variables. Le graphe d'un CSP binaire est appelé graphe des contraintes.

On peut répartir les méthodes de résolution des CSP en deux grandes familles :

- Les méthodes déterministes où l'on instancie une à une les variables du CSP en revenant sur des choix d'assignement si l'on aboutit à un viol des contraintes.
- Les méthodes stochastiques où l'on part d'un assignement complet (c'est-à-dire un assignement de toutes les variables) et où l'on modifie celui-ci de telle sorte que l'on se rapproche d'une solution du CSP.

4.3.2 Les méthodes de résolution déterministes.

Les méthodes déterministes sont basées sur le *backtracking*. Dans sa version élémentaire, appelée parfois *chronological backtracking*, on instancie les variables les unes après les autres. Quand on arrive à une impasse (impossibilité d'affecter une valeur à une nouvelle variable sans que l'on soit en conflit avec au moins une contrainte), on revient sur la précédente assignation de variable. Dans cet algorithme, l'ordre d'instanciation des variables et d'exploration des domaines est fixé au préalable, de façon purement conventionnelle. On notera que la complexité algorithmique du *backtracking* est exponentielle.

Une des améliorations possible du *backtracking* consiste à choisir de façon judicieuse (et non plus automatique) la variable que l'on va instancier. Il est aussi intéressant de choisir avec attention la valeur du domaine que l'on va assigner à la variable retenue. Les ordres

choisis pour les variables et les valeurs peuvent être statiques (déterminés a priori une fois pour toute) ou dynamiques.

Par exemple, l'heuristique du *fail-first principle* sélectionne la variable ayant le plus petit nombre de valeurs satisfaisantes (pour les contraintes) dans son domaine. L'idée de cette heuristique est de détecter le plus rapidement possible des contradictions dans des assignements partiels afin d'éviter des explorations inutiles de l'arbre de recherche associé au processus de *backtracking*. Une autre heuristique possible consiste à prendre la variable apparaissant dans le plus grand nombre de contraintes.

Au niveau de l'ordre des valeurs choisies pour une variable, la technique *lookahead value ordering* calcule le nombre de fois où une valeur est en conflit avec une valeur d'une variable restant à instancier. C'est la valeur générant le moins de conflits que l'on sélectionnera alors.

Une autre amélioration du *backtracking* revient à prendre en compte l'impact d'une assignation de variable sur les variables non encore instancieés. On parle de propagation de contraintes et la notion centrale de ces algorithmes est celle de consistance d'arc.

On dit que l'arc (X_i, X_j) est consistant si $\forall u \in D(X_i)\ \exists v \in D(X_j)\ (u,v)$ satisfait les contraintes portant sur les variables X_i et X_j. $D(X_i)$ correspond au domaine courant associé à la variable X_i (Au cours des instanciations successives des variables, des valeurs peuvent être retirées de ce domaine du fait de leur incompatibilité avec l'assignation partielle courante).

Le graphe des contraintes du CSP sera arc-consistant si tous ses arcs sont consistants.

Rendre l'arc (X_i, X_j) consistant revient à retirer de $D(X_i)$ les valeurs ne pouvant remplir les contraintes associées aux variables X_i et X_j. On peut aussi concevoir cette action comme un ajout de contraintes (qui étaient implicites) au CSP.

Remarquons que le fait de rendre le graphe des contraintes arc-consistant ne signifie pas qu'il suffira ensuite de choisir arbitrairement des valeurs pour chaque variable dans les domaines pour avoir une solution du CSP. On peut même avoir un graphe arc-consistant et aucune solution pour le CSP associé. Cependant, on démontre que si le graphe des contraintes présente une structure d'arbre, alors la consistance d'arcs et la consistance de nœuds (qui revient à écarter d'un domaine les valeurs en contradiction avec une contrainte unaire portant sur la variable de ce domaine) permettent d'obtenir une solution sans *backtracking*.

Les algorithmes reposant sur cette notion de consistance d'arc se différencient par le fait qu'ils garantissent l'arc-consistance du graphe ou d'une partie de celui-ci (voire d'un simple filtrage de valeurs dans des domaines pour le *forward checking*), que l'arc-consistance est assurée à chaque étape du *backtracking* ou au début, et finalement par la qualité des algorithmes chargés de rendre le graphe des contraintes arc-consistant.

Par exemple, le *forward checking*, après l'instanciation d'une variable X, effectue pour toutes les variables Y non encore instanciées une réduction de leurs domaines pour conserver seulement les valeurs compatibles avec la valeur de X choisie. Très naturellement, un algorithme de *forward checking* utilise l'heuristique du *fail-first principle* après la phase de filtrage des domaines que nous venons de décrire.

Les algorithmes de type *lookahead* appliquent le filtrage des valeurs du *forward checking* et par ailleurs assurent la consistance d'arcs entre les variables non encore instanciées.

Finalement, les algorithmes MAC (*Maintaining Arc Consistency*) assurent l'arc-consistance du graphe à chaque étape du *backtracking*. Ils commencent par établir préalablement l'arc-consistance du graphe puis la conservent à chaque assignation.

Ces algorithmes aboutissent donc à des réductions de domaines de plus en plus importantes, au prix de calculs pour établir la consistance d'arcs.

Pour assurer l'arc-consistance du graphe des contraintes, on dispose de procédés plus ou moins efficaces : de AC-1 à AC-7.

Les premiers algorithmes AC recourent à une fonction appelée couramment *revise* dans les articles consacrés à la consistance d'arcs. Pour un arc orienté (X_i, X_j), cette fonction supprime du domaine de X_i les valeurs ne pouvant satisfaire les contraintes portant sur X_i et X_j.

Fonction revise(X_i,X_j) retourne booléen
    suppression ← faux
    Pour tout $u \in D(X_i)$ faire
        Si $\exists v \in D(X_j)$ tel que $(u,v) \in C(X_i, X_j)$ alors
            $D(X_i) \leftarrow D(X_i) - \{u\}$
            suppression ← vrai
        Fin si
    Fin pour tout
    retourner suppression
Fin Fonction

On remarque que quand on applique $revise(X_i, X_j)$ on peut supprimer de $D(X_i)$ des valeurs qui servaient pour la consistance des arcs (X_k, X_i). Ceci oblige donc à revoir la consistance de (X_k, X_i) pour $k \in \{1, \cdots, n\} - \{i, j\}$.

L'algorithme AC-1 applique *revise* à tous les arcs (X_i, X_j) jusqu'à ce qu'il n'y ait plus aucune modification des domaines des variables. Il se présente donc sous la forme suivante :

Procédure AC-1
    Répéter
        Changement ←faux
        Pour i de 1 à n faire
            Pour j de 1 à n faire
                Si i≠j et $\exists C(X_i, X_j)$ alors
                    Changement← revise(X_i,X_j) ou changement
                Fin si
            Fin pour
        Fin pour
    Jusqu'à ce que non(changement)
Fin Procédure

L'algorithme AC-3 est une amélioration de AC-1. Il tient à jour une liste L des arcs à voir ou à revoir jusqu'à obtenir l'arc-consistance du CSP. Une version classique de cet algorithme est :

Procédure AC-3
    L ← $\{(X_i, X_j) / i \neq j \text{ et } \exists C(X_i, X_j)\}$
    Tant que $L \neq \emptyset$ faire

Choisir un arc (X_k, X_m) dans L
L ← L – $\{(X_k, X_m)\}$
Si revise (X_k, X_m) alors
L ← $L \cup \{(X_i, X_k) / \exists C(X_i, X_k)\ et\ i \neq k\ et\ i \neq m\}$
Fin si
Fin tant que
Fin Procédure

Une autre famille d'algorithmes cherchant à améliorer les performances du *backtracking* a pour but d'éviter de remettre systématiquement en cause la dernière assignation effectuée quand on aboutit à une situation d'échec dans l'exploration. En effet, ce n'est pas nécessairement au niveau de la dernière variable instanciée que peut résider la source du problème rencontré. Aussi, avant de revenir sur l'assignation qui est véritablement responsable de la situation d'échec, le *backtracking* va faire des explorations inutiles (car conduisant toutes finalement à des violations de contraintes).

Considérons par exemple l'instanciation de variables suivante : $X_1,\ldots,X_i,\ldots,X_j$. On suppose que cette instanciation est consistante (c'est-à-dire respecte les contraintes entre les variables). On essaie ensuite d'assigner une valeur à une variable X_k du problème. On trouve qu'aucune valeur de X_k ne permet de satisfaire une contrainte entre X_i et X_k. On dit alors que X_i est une variable de conflit. Un *backtracking* simple amène à changer la valeur de la variable précédemment instanciée, X_j, alors que l'algorithme *backjumping* remettra en cause la valeur courante de X_i qui est la source de l'échec rencontré (ceci en effectuant un retour-arrière au niveau de X_i et non pas de X_j dans l'arbre d'exploration). On notera que si au moment de l'assignation de la valeur à X_i aucune valeur du domaine de X_k ne satisfait la contrainte entre X_i et X_k , un algorithme comme *forward checking* détectera tout de suite ce conflit. De façon générale, on établit que le *backjumping* n'est d'aucune utilité associé au *forward checking* où à un algorithme garantissant une consistance plus forte comme MAC.

L'algorithme *conflict-directed backjumping* est une version évoluée du *backjumping*. L'ensemble des variables de conflit d'une variable X_i n'est plus simplement la totalité des variables instanciées partageant une contrainte avec X_i ; c'est l'ensemble des variables instanciées qui empêchent X_i et d'autres variables non encore instanciées de constituer une instanciation consistante.

Une autre amélioration de ce type d'algorithmes consiste à mémoriser les instanciations inconsistantes (*nogoods*) afin d'éviter des assignations inutiles (qui sont des répétitions d'échecs survenus auparavant). Dans l'exemple illustrant le principe du *backjumping*, la valeur associée à X_i, conduisant à un échec, est un *nogood*. Ainsi, si l'on explore par la suite de nouvelles branches pour les variables $X_1,\ldots,X_{i-1}$, la mémorisation du *nogood* permettra de ne pas assigner à X_i la valeur conduisant à un conflit avec X_k.

4.3.3 Les méthodes de résolution stochastiques.

Le principe général de ces méthodes est d'affecter aléatoirement une valeur à chaque variable du CSP puis de réparer l'instanciation globale de telle sorte qu'elle finisse par satisfaire la totalité des contraintes. On cherche donc à minimiser le nombre de contraintes non satisfaites par des modifications successives de l'instanciation initiale. Dans ce processus, le risque est d'aboutir à des minima locaux et de ne pouvoir parvenir à une solution du CSP.

L'algorithme de *hill-climbing* consiste à examiner l'ensemble des instanciations voisines de l'instanciation globale courante. Les instanciations voisines sont celles qui diffèrent seulement quant à la valeur d'une variable. Parmi les instanciations voisines, on recherche

celle(s) qui minimise(nt) le nombre de contraintes non respectées. Si on peut diminuer le nombre de contraintes non satisfaites, on choisit alors cette solution (S'il y a plusieurs instanciations voisines candidates, on fait un choix aléatoire). Quand le *hill-climbing* parvient à un minimum local (Aucun changement sur aucune variable ne permet de diminuer le nombre de contraintes non satisfaites), l'algorithme choisit une nouvelle instanciation globale de manière stochastique.

On notera que le nombre d'instanciations voisines de l'instanciation courante examinées par cette méthode est élevé (Il vaut n(m-1) où n est le nombre de variables du CSP et m la taille moyenne d'un domaine d'une variable).

Afin de réduire la taille de l'ensemble des instanciations évaluées à chaque étape, l'algorithme *min-conflicts* choisit la variable dont on modifie la valeur parmi les variables dont l'assignement actuel entraîne le non respect d'au moins une contrainte. Une fois cette variable choisie, on lui affecte la valeur qui minimise le nombre de contraintes violées par l'instanciation globale. Si une telle valeur n'existe pas, on sélectionne aléatoirement une valeur qui n'augmente pas le nombre courant de contraintes violées. Au bout d'un certain nombre d'étapes, si l'on n'a trouvé aucune solution, l'algorithme retourne un indicateur d'échec (avec éventuellement la solution approchée obtenue à l'issue du processus d'exploration).

On remarquera que la méthode *min-conflicts* peut rester bloquée dans un minimum local. On pallie à cela avec des stratégies de type *random-walk* où l'on introduit une perturbation aléatoire dans le fonctionnement de l'algorithme *min-conflicts*.

La méthode de recuit simulé a aussi été adaptée aux CSP. A chaque étape, une instanciation voisine de l'instanciation globale courante est engendrée aléatoirement. Les instanciations voisines correspondent aux instanciations globales différant de l'instanciation courante uniquement par la valeur d'une variable en conflit.

Si la nouvelle instanciation diminue le nombre de contraintes non satisfaites, elle est systématiquement choisie et se substitue à l'instanciation actuelle.

Sinon, appelons Δ la différence entre le nombre de contraintes non satisfaites de la nouvelle instanciation et le nombre de contraintes non vérifiées avec l'instanciation actuelle. On a : $\Delta > 0$. L'algorithme de recuit simulé décide alors de sélectionner la nouvelle instanciation globale (bien qu'elle dégrade la qualité de l'instanciation) avec une probabilité définie par la fonction $f(\Delta, T) = e^{-\frac{\Delta}{T}}$. Observons l'évolution de f selon chacun de ses paramètres Δ et T. Soit $g(\Delta) = e^{-\frac{\Delta}{T}}$. Sur $[0,+\infty[$, g décroît strictement de 1 à 0. Donc les dégradations les plus faibles sont les mieux acceptées puisque la probabilité f est alors proche de 1. Etudions maintenant le paramètre T que l'on assimile classiquement à une température. Soit $h(T) = e^{-\frac{\Delta}{T}}$. Sur $[0,+\infty[$, h croît strictement de 0 à 1. Donc plus la température est élevée, plus on a de probabilité d'accepter une dégradation de la qualité de l'instanciation globale. Le paramètre T est contrôlé par une fonction décroissante au cours du temps, fonction définissant ce que l'on nomme le schéma de refroidissement. La performance de l'algorithme dépend fortement du schéma de refroidissement choisi (On a souvent une réduction de T par paliers).

L'algorithme s'arrête quand :

- soit on a obtenu une solution,
- soit aucune instanciation voisine n'a été acceptée pendant un certain nombre d'itérations pour une valeur T fixe,
- soit T a atteint la valeur 0,

- soit on a atteint un nombre maximum de changements (valeur qui est prédéfinie).

L'algorithme peut donc s'arrêter avec une instanciation globalement inconsistante. Il renvoie alors la meilleure instanciation qu'il a trouvée.

Terminons cet inventaire des méthodes de résolution stochastiques des CSP par la recherche tabou. A chaque étape, la méthode tabou examine la totalité des instanciations voisines (sauf certaines appartenant à une liste tabou dont nous donnons la signification par la suite) de l'instanciation globale actuelle. Les instanciations voisines sont définies de la même manière que pour le recuit simulé.

On choisit de sélectionner la meilleure instanciation voisine, même si elle est de moins bonne qualité que l'instanciation courante. Ceci permet de ne pas rester bloqué dans un minimum local. De plus, afin d'éviter de cycler sur un ensemble d'instanciations, l'algorithme tient à jour une liste d'instanciations taboues qu'il n'examinera pas lors de la phase de recherche d'une nouvelle instanciation. Lors d'un changement d'instanciation, l'ancienne instanciation est rajoutée à la liste tabou. La liste tabou a souvent une taille fixée a priori.

L'algorithme s'arrête quand :

- soit on a obtenu une solution,
- soit on a accompli un certain nombre d'étapes et l'algorithme renvoie alors la meilleure instanciation qu'il a obtenu lors de son exploration.

Comme pour le recuit simulé, l'algorithme peut s'achever en fournissant une instanciation qui n'est pas globalement consistante (c'est-à-dire qui n'est pas solution du CSP).

4.4 Problèmes rencontrés avec les CSP obtenus et analyse de ceux-ci.

Nous avons introduit dans le paragraphe 4.2 une version modifiée de notre algorithme pour faciliter la résolution des CSP exprimant les objectifs spécifiés par le concepteur. Toutefois, nous allons voir que cet ajustement se révèle insuffisant. Ceci est dû à la nature même des CSP qui sont produits avec l'approche choisie.

Remarquons par ailleurs que les méthodes stochastiques de résolution d'un CSP permettent d'obtenir une solution (parfois simplement une solution approchée), mais ne sont pas adaptées à la recherche de toutes les solutions d'un CSP. Elles ne conviennent donc pas, de par leur essence, à notre problème.

Examinons la structure des CSP que nous avons à résoudre.

Appelons, comme au paragraphe 4.2, L la liste des carreaux pouvant constituer la source de lumière.

Nous commencerons pour notre analyse des CSP avec une source lumineuse de forme carrée.

Soit S la taille de la source lumineuse carrée que nous devons introduire dans la scène pour satisfaire les spécifications fournies par l'utilisateur. S est le nombre de carreaux du nouveau maillage formant la source de lumière. La quantité S décrit donc un ensemble de valeurs consécutives des entiers naturels.

Soit (X,Y) la coordonnée du carreau situé dans le haut gauche de la source de lumière. Les domaines de X et Y décrivent la largeur et la longueur de la zone correspondant à L.

Les contraintes liées au placement de la source lumineuse sont alors :

$$\begin{cases} (X,Y)\in L & \dots & (X,Y+S-1)\in L \\ (X+1,Y)\in L & \dots & (X+1,Y+S-1)\in L \\ \dots & \dots & \dots \\ (X+S-1,Y)\in L & \dots & (X+S-1,Y+S-1)\in L \end{cases}$$

La formulation du problème fait que nous avons trois variables et S^2 contraintes entre celles-ci.

Si nous fixons S, nous n'avons plus que deux variables et le graphe des contraintes est alors un multi-graphe. Toutefois, il est alors possible d'exprimer toutes les contraintes en extension et nous pouvons nous ramener à une seule contrainte entre les variables X et Y par des intersections.

En effet, on a : $(X+I,Y+J)\in L \Leftrightarrow (X,Y)\in L_{I,J}$ avec :

$$L_{I,J} = \{(x-I,y-J) / (x,y)\in L\} \cap (D_X \times D_Y).$$

Donc le CSP se ramène à :

$$(X,Y)\in \bigcap_{(I,J)\in ([0,S-1]\cap N)^2} L_{I,J}$$

Nous avons supposé dans un premier temps S fixé. Or la taille de la source lumineuse est bien entendu susceptible d'évoluer (la description de la taille pouvant être formulée de manière imprécise). Cependant, quand la valeur de S change, c'est un nouveau CSP que nous obtenons (avec un nombre différent de contraintes). Nous avons ainsi à résoudre une suite de CSP.

Cette tâche est assez simple si nous partons de l'assignation S=1 et incrémentons S. En effet, en incrémentant S, pour trouver les solutions du nouveau CSP, il suffit de supprimer des couples solutions de l'étape précédente avec l'introduction de nouveaux ensembles $L_{I,J}$ dans l'intersection fournissant les couples (X,Y) solutions.

On voit qu'avec un ordre particulier d'instanciation des variables (S, puis X et Y simultanément), la résolution des CSP rencontrés ne fait appel à aucune des techniques présentées dans le chapitre 4.3.

Examinons maintenant le cas de deux sources lumineuses de forme carrée.

Les variables X, Y et S gardent leur signification précédente ; elles sont désormais des caractéristiques de la première source de lumière.

On notera X', Y' et S' les variables spécifiant complètement la seconde source de lumière.

On a les contraintes suivantes correspondant au fait que les sources lumineuses sont situées dans la zone décrite par L :

$$\begin{cases} (X,Y)\in L & \dots & (X,Y+S-1)\in L \\ \dots & \dots & \dots \\ (X+S-1,Y)\in L & \dots & (X+S-1,Y+S-1)\in L \\ (X',Y')\in L & \dots & (X',Y'+S'-1)\in L \\ \dots & \dots & \dots \\ (X'+S'-1,Y')\in L & \dots & (X'+S'-1,Y'+S'-1)\in L \end{cases}$$

Par ailleurs, la seconde source lumineuse ne doit pas posséder de carreaux en commun avec la première. Notons $L_{(X,Y)}$ l'ensemble des carreaux de la première source de lumière. Nous avons :

$$L_{(X,Y)} = \{(X,Y),\dots,(X+S-1,Y),\dots,(X,Y+S-1),\dots,(X+S-1,Y+S-1)\}$$

La condition de non intersection des sources lumineuses se traduit par les contraintes suivantes :

$$\begin{cases} (X',Y') \notin L_{(X,Y)} & \dots & (X',Y'+S'-1) \notin L_{(X,Y)} \\ \dots & \dots & \dots \\ (X'+S'-1,Y') \notin L_{(X,Y)} & \dots & (X'+S'-1,Y'+S'-1) \notin L_{(X,Y)} \end{cases}$$

Comme précédemment, nous allons d'abord fixer les tailles S et S' des sources lumineuses, c'est-à-dire instancier en premier ces variables du CSP. Les contraintes liées au positionnement des sources de lumière dans L se traitent alors par des intersections d'ensembles finis comme nous l'avons fait pour le cas d'une seule source de lumineuse (Chaque contrainte pouvant aisément s'exprimer en extension). On se ramène donc à deux contraintes :

$$\begin{cases} (X,Y) \in L_S \\ (X',Y') \in L_{S'} \end{cases}$$

où les ensembles L_S et $L_{S'}$ sont calculés comme nous venons de l'expliquer.

Il reste donc à prendre en compte les contraintes garantissant le non recouvrement des sources de lumière. On remarque que ces contraintes ne sont pas binaires mais font intervenir 4 variables : X, Y, X' et Y'.

Pour résoudre le CSP, on va à présent instancier simultanément les variables X et Y en choisissant un couple de L_S. Une fois cette opération accomplie, les éléments de l'ensemble $L_{(X,Y)}$ sont tous connus. On écrit alors les contraintes en extension grâce au procédé employé auparavant, à savoir :

$$(X'+I,Y'+J) \notin L_{(X,Y)} \Leftrightarrow (X',Y') \notin \left(\{(x-I,y-J)/(x,y) \in L_{(X,Y)}\} \cap (D_{X'} \times D_{Y'}) \right)$$

Par intersection, on aboutit à une contrainte de la forme :

$$(X',Y') \notin E_{(X,Y)}$$

Et, en faisant finalement intervenir la contrainte de placement pour la seconde source lumineuse, on obtient :

$$(X',Y') \in L_{S'} - E_{(X,Y)}$$

On obtient donc pour (X',Y') tous les couples solutions une fois que S, S', X et Y ont été fixés.

On remarquera que l'extension de la technique que nous venons d'exposer à des sources lumineuses rectangulaires ainsi qu'à un nombre de sources lumineuses supérieur à 2 est triviale.

On voit que la méthode que nous proposons est une technique de résolution ad hoc qui a peu recours aux principes employés dans les méthodes classiques de résolution des CSP (quoiqu'elle ait quelque ressemblance avec le *forward checking*). En fait, bien que notre problème puisse se modéliser comme une suite de CSP à résoudre, la recherche des solutions n'utilise pas un algorithme usuel de résolution d'un CSP. La variable S influe directement sur le nombre de contraintes du CSP. Quant à X et Y, ces variables sont instanciées simultanément car un traitement individuel de celles-ci compliquerait sévèrement la résolution.

Il reste que nous obtenons au final un nombre élevé de scènes solutions. En effet, l'émittance attribuée aux carreaux constituant les sources de lumière décrit un intervalle de valeurs (puisque l'on en a une description déclarative). Le nombre de sources lumineuses peut lui aussi être imprécis, ce qui fait que toutes les configurations possibles pour un nombre valide de lumières sont ajoutées à l'ensemble des solutions. On peut faire la même remarque

pour la taille d'une source lumineuse. Par ailleurs, notons que si le concepteur ne donne aucune propriété portant sur l'un de ces paramètres, nous devons supposer que celui-ci parcourt l'ensemble de ses valeurs possibles, situation où l'on aura donc le nombre maximal de solutions liées à ce paramètre.

La gestion de l'ensemble des solutions apparaît donc comme un aspect critique du problème d'éclairage inverse présenté sous forme de spécifications déclaratives.

Finalement, deux types de situations se présentent pour un problème d'éclairage inverse formulé de façon déclarative :

- soit on a une grande quantité de solutions (comme dans la scène que nous avons étudiée) et il n'est pas vraiment utile de recourir à une formulation à base de CSP (qui aboutit à des calculs d'intersections pour une optimisation des solutions peu utile en réalité),
- soit les objectifs donnés par l'utilisateur sont contradictoires et il n'y a aucune solution.

Cette remarque nous amène à proposer une autre méthode. Puisque quand les buts exigés par le concepteur peuvent être remplis il y a une grande quantité de solutions possibles, il paraît plus simple d'engendrer de façon systématique toutes les scènes qui correspondent plus ou moins aux spécifications demandées. Toute la difficulté est donc reportée sur la gestion de cet ensemble de scènes solutions. Nous avons indiqué que jusqu'à présent la phase de prise de connaissance se limitait à la visualisation des scènes satisfaisant les spécifications fournies par l'utilisateur. Avec un nombre élevé de scènes solutions, il est nécessaire de réfléchir à l'introduction de classements et de présentations raisonnées de ces scènes. Le problème auquel nous a conduit la prise en compte de la nature déclarative des spécifications d'éclairage relève ainsi du domaine de la visualisation de l'information.

5 Classification et représentation des scènes solutions.

5.1 Introduction.

Dans les paragraphes précédents, notre étude sur l'éclairage inverse direct à base de spécifications déclaratives nous a montré le peu d'utilité d'une modélisation avec des CSP (méthode couramment utilisée dans le domaine de la modélisation déclarative).

Les raisons profondes expliquant cette situation proviennent de la nature même des problèmes d'éclairage inverse.

Pour une spécification d'éclairage de la scène, on a :

- soit aucune solution (Parce que l'on a exigé une radiosité bien précise pour un carreau et une seconde radiosité de façon aussi précise pour un carreau voisin, ou alors simplement parce que l'on a une contradiction entre les objectifs).
- soit un nombre très élevé de solutions (En demandant seulement une radiosité pour un carreau P par exemple : il y a alors de multiples possibilités pour placer un carreau faisant office de source lumineuse et éclairant correctement le carreau P).

Dans notre état de l'art sur l'éclairage inverse, nous avons noté le grand nombre de méthodes d'optimisation employées. Comme, dans ces articles, les spécifications se font de manière précise et numérique (en peignant des carreaux de la scène par exemple), il n'y a aucune solution (mathématiquement correcte) au problème d'éclairage inverse introduit par l'utilisateur. Des méthodes d'optimisation sont donc nécessaires afin de se rapprocher le plus possible des spécifications données.

Avec notre approche, étant donné que les surfaces sont lambertiennes et que les objectifs sont spécifiés sous forme déclarative (ce qui se traduit numériquement par des intervalles), on se retrouve souvent dans le cas où il y a énormément de solutions. Si l'on se place dans le cadre de l'illumination globale (et plus simplement avec un éclairage direct), on

aura encore plus de solutions possibles. Notre problème consiste donc essentiellement à permettre à l'utilisateur de pouvoir exploiter convenablement cet ensemble de scènes solutions.

5.2 Utilisation de techniques de visualisation de l'information pour gérer un nombre élevé de solutions.

5.2.1 Principes de la méthode.

Une solution au problème rencontré est d'engendrer uniquement des scènes intéressantes (c'est-à-dire suffisamment différentes les unes des autres) puis d'effectuer un classement de celle-ci pour les présenter de façon rationnelle à l'utilisateur.

Pour reprendre le vocabulaire de la modélisation déclarative, l'idée fondamentale est de s'appuyer principalement sur la phase de prise de connaissance.

L'approche que nous proposons est similaire à celle de [MAR97]. Toutefois, la méthode de cet article pouvant s'appliquer potentiellement à n'importe quel domaine de l'infographie, en faisant intervenir les propriétés spécifiques du problème que nous étudions, nous allons pouvoir pallier à quelques défauts des techniques utilisées dans [MAR97].

Avant de détailler notre approche, nous allons présenter les travaux de [MAR97] dont notre approche s'inspire tout en s'en démarquant sur plusieurs points.

5.2.2 Les *Design Galleries*.

5.2.2.1 Présentation générale.

[MAR97] ont présenté les *Design Galleries* comme un nouveau paradigme pour la sélection de paramètres adéquats en infographie. Cette méthode se veut assez généraliste pour pouvoir aborder des problèmes comme l'éclairage d'une scène, les systèmes de particules, l'animation d'objets, etc.

L'idée de fondamentale est d'engendrer de façon automatique un ensemble de solutions (c'est-à-dire des affectations de valeurs aux paramètres du problème) assez différentes entre elles, de sorte que l'on présente à l'utilisateur une collection représentative des solutions possibles de son problème. L'exploration de l'espace des solutions est accomplie dans une phase de pré-calculs.

[MAR97] considèrent le fait de générer un ensemble significatif de l'espace solution comme un avantage par rapport à une méthode inverse. Ils remarquent que souvent le concepteur n'a pas une idée bien définie des effets qu'il souhaite produire, et donc une difficulté à simplement formuler ceux-ci.

Les constituants principaux d'un système de *Design Gallery* sont :

- Le vecteur d'entrée, formé des paramètres du problème.
- Le vecteur de sortie, constitué des quantités qui caractérisent de façon fondamentale les solutions.
- Une distance sur l'espace des vecteurs de sortie, distance servant à mesurer la ressemblance entre deux solutions.
- Un algorithme de dispersion chargé de trouver une suite de vecteurs d'entrée qui vont produire des solutions suffisamment différentes, aboutissant ainsi à la formation d'un ensemble de solutions qualifié de « bien distribué ».
- Un algorithme d'arrangement des solutions servant à présenter de manière organisée l'ensemble bien distribué des solutions.

Plusieurs algorithmes de dispersion et d'arrangement sont proposés dans l'article. Ils se veulent généralistes (au sens où ils s'appuient uniquement sur la distance entre deux vecteurs solutions).

Ainsi, pour mettre en place un nouveau système de *Design Gallery*, il suffit de préciser :

- Les composantes du vecteur d'entrée.
- Les composantes du vecteur de sortie (et la fonction permettant de calculer un vecteur de sortie à partir d'un vecteur d'entrée).
- Une métrique sur l'ensemble des vecteurs de sortie.

[MAR97] ont donné plusieurs exemples issus de leur paradigme. Nous allons ici nous concentrer sur leur *Design Gallery* adaptée à l'éclairage d'une scène.

Dans ce cas, un vecteur d'entrée est constitué d'un ensemble de paramètres dépendant de la nature des sources de lumière intervenant (source ponctuelle, source polygonale, spot ou lumière directionnelle). Pour une source ponctuelle, une source polygonale ou un spot, la position peut être restreinte à un sous-ensemble des surfaces de la scène. Pour une lumière directionnelle, des points devant être éclairés par celles-ci sont choisis dans différents endroits de la scène.

Les vecteurs de sortie sont une suite d'images de taille diverses (Une résolution de 32x25 pixels étant la résolution la plus élevée pour ces images). Ces images sont le résultat d'un processus de rendu arbitraire (lancer de rayons, par exemple).

La distance choisie prend en compte les images produites (c'est-à-dire les vecteurs de sortie) en pondérant leur influence. Cette distance peut être associée à une norme d'un espace L^1.

5.2.2.2 Deux algorithmes de dispersion.

Le premier algorithme de dispersion présenté par [MAR97] consiste à choisir progressivement dans l'ensemble global des solutions celles qui sont les plus convenables pour la construction de l'ensemble bien équilibré.

Dans un premier temps, toutes les solutions sont engendrées dans un ensemble que nous appellerons E.

Notons I l'ensemble bien équilibré des solutions.

Au départ, on a : $I = \varnothing$.

A chaque étape de l'algorithme, on détermine une solution qui va être ajoutée à l'ensemble I.

Une étape est composée des opérations suivantes :

- Trouver $x_0 \in E$ tel que $\max_{x \in E} \min_{y \in I} d(x, y) = \min_{y \in I} d(x_0, y)$ (d est la distance entre deux solutions)
- $I \leftarrow I \cup \{x_0\}$
- $E \leftarrow E - \{x_0\}$

Le nombre d'étapes est fixé à l'avance. Il dépend de la méthode d'arrangement. En effet, il s'agit de générer un nombre suffisamment important de solutions tout en restant dans une limite permettant de représenter de façon satisfaisante l'ensemble bien équilibré I que l'on obtient.

L'algorithme proposé a comme défaut le fait de commencer par construire globalement l'ensemble des solutions sans prendre en compte les informations sur les solutions obtenues progressivement lors de cette génération. [MAR97] proposent un deuxième algorithme de dispersion qui évite d'engendrer la totalité des solutions avant d'effectuer un filtrage.

Le deuxième algorithme est assez similaire dans son esprit à un algorithme génétique.

On commence par initialiser I avec une suite de vecteurs d'entrée aléatoires. La cardinalité de I est invariante tout au long de l'algorithme.

A chaque étape de l'algorithme, on modifie légèrement et de façon aléatoire un vecteur de I. Si le nouveau vecteur obtenu permet de constituer un ensemble mieux équilibré, alors il remplace dans I un vecteur qui produisait un ensemble moins intéressant. La substitution de vecteur a lieu s'il est possible d'augmenter de la sorte la quantité $\min_{(u,v)\in O^2} d(u,v)$ (où O est l'ensemble des vecteurs de sortie associé à l'ensemble I des vecteurs d'entrée). De façon intuitive, augmenter la quantité précédente revient à « faire s'éloigner les uns des autres » les vecteurs de sortie.

5.2.2.3 Deux algorithmes d'arrangement.

L'ensemble des solutions bien équilibré I possède une cardinalité élevée. Ainsi se pose naturellement la question de la représentation de cet ensemble pour que l'utilisateur puisse en saisir la structure générale et les caractéristiques des éléments qui l'intéressent plus spécifiquement. C'est la phase d'arrangement des solutions qui prend en charge la visualisation de l'information contenue dans l'ensemble bien équilibré.

[MAR97] proposent une première méthode où les solutions sont rangées dans une forêt. Chaque arbre de la forêt est censé correspondre à des configurations où l'effet d'éclairage de la scène est semblable.

Pour construire cette collection d'arbres, [MAR97] commencent par constituer un graphe complet où les nœuds représentent les solutions et le poids d'une arête (x_i,x_j) est $\frac{1}{d(x_i,x_j)}$.

Un k-partitionnement du graphe est ensuite effectué. Cette opération consiste à construire des sous-graphes d'ordre k tels que la somme des arêtes supprimées (arêtes reliant des sommets de sous-graphes distincts) soit minimale. Ainsi, on a tendance à regrouper dans les sous-graphes des solutions similaires du point de vue de la distance (avec les poids choisis pour les arêtes).

Dans chaque sous-graphe, une solution est ensuite sélectionnée. Elle sert de racine à un arbre que l'on construit en parcourant le sous-graphe.

Le nombre d'arbres (c'est-à-dire la constante k) et la hauteur h de chacun des arbres sont choisies de telle sorte que l'affichage des nœuds de la forêt soit suffisamment clair pour que l'utilisateur puisse distinguer des différences entre des solutions éloignées. Si l'on choisit des valeurs trop grandes pour k et h, les images seront illisibles à l'écran. [MAR97] ont choisi k = 8 et h = 2. Donc, avec ce choix, le nombre de solutions dans l'ensemble bien équilibré est de 584.

La représentation de la forêt choisie par [MAR97] est peu pratique (Elle est partielle et reflète très mal la structure d'arbre). De plus, la méthode d'arrangement introduite conduit fréquemment au regroupement dans un même arbre de scènes dont l'éclairage est sensiblement différent. Ceci provient du k-partitionnement accompli lors de la phase d'arrangement (qui ne s'appuie sur aucune information sémantique) et du fait que la structure de l'ensemble solution a souvent du mal à être représentée par une forêt.

Ceci a amené les auteurs à rechercher une autre technique de visualisation des scènes solutions. Cette méthode est une méthode statistique : le MDS (*Multidimensional Scaling*). A partir d'une matrice de distances entre différents vecteurs de $\Re^n$, la méthode MDS calcule les coordonnées de points de $\Re^m$ (avec m<n) tels que la distance euclidienne entre ces points

corresponde à la distance entre les vecteurs associés. Classiquement, m = 2 (représentation dans le plan) ou m = 3 (représentation dans l'espace). Les points que nous venons d'évoquer correspondent dans le cas de [MAR97] à des images des scènes solutions dans une échelle assez petite. Dans la pratique, avec une représentation 2D, les images ont tendance à se recouvrir (Problème de *cluttering* pour reprendre le vocabulaire du domaine de la visualisation de l'information.). Ce phénomène est la conséquence de la perte d'information due à la projection d'un espace de dimension n avec n>>2 (puisque cet espace correspond à l'ensemble des solutions) sur un espace de dimension 2.

5.2.2.4 Résultats obtenus.

Sur la figure 55 apparaît le résultat obtenu pour un problème d'éclairage avec la première méthode d'arrangement (construction d'une forêt à partir de l'ensemble des solutions).

La première rangée d'images en haut de l'écran correspond aux racines de chacun des arbres de la forêt. Une racine a été sélectionnée (Son cadre est surligné en blanc). La deuxième rangée contient les nœuds fils de cette racine. Une solution a aussi été choisie à ce niveau. La troisième rangée est remplie par les nœuds fils solutions de la solution sélectionné au deuxième niveau.

L'image dans le bas gauche est une composition des images situées à sa droite. Ces dernières ont été choisies par l'utilisateur qui aura parcouru les différents arbres de solutions en sélectionnant des nœuds.

On remarque que :

- La forêt n'est pas visible dans sa totalité (Le concepteur a toujours une vue partielle de l'ensemble solution).
- L'arbre visualisé partiellement regroupe des images dont l'éclairage diffère fortement. Certaines images seraient mieux placées dans un autre arbre.

La figure 56 illustre l'utilisation de la méthode MDS pour un problème de choix des paramètres dans un système de particules.

L'ensemble des solutions est représenté au centre de l'écran.

Le problème de recouvrement (*cluttering*) entre les différentes images correspondant à une solution apparaît clairement.

L'interface permet à l'utilisateur de se déplacer et de zoomer à l'intérieur de l'ensemble des solutions. L'utilisateur peut ainsi « naviguer » dans l'ensemble des solutions à la recherche de celles qu'il va juger le mieux correspondre à l'ambiance lumineuse qu'il désire obtenir. Ceci n'élimine cependant pas le problème de la perception de la structure globale de l'ensemble solution. Du fait du recouvrement de certaines images par d'autres, il peut être difficile pour l'utilisateur de choisir l'endroit où il va devoir zoomer pour obtenir une solution intéressante.

[AND97], suivant les travaux de [MAR97], proposent une application de cette méthode de représentation des solutions au problème de l'éclairage d'une scène. Sur la figure 57 est visualisé un ensemble de 584 images solutions (obtenues par lancer de rayons). Il y a une seule source de lumière dans la scène ; seuls son type et ses paramètres varient.

5.2.2.5 L'utilisation de la 3D pour la présentation des solutions.

[AND97] ont utilisé la méthode MDS pour représenter le graphe complet des solutions, non plus en 2D mais en 3D. Les solutions sont modélisées par des images placées dans un monde virtuel (en VRML).

Figure 55. ***Design Gallery*** **pour l'éclairage d'une scène (Image issue de [MAR97]).**

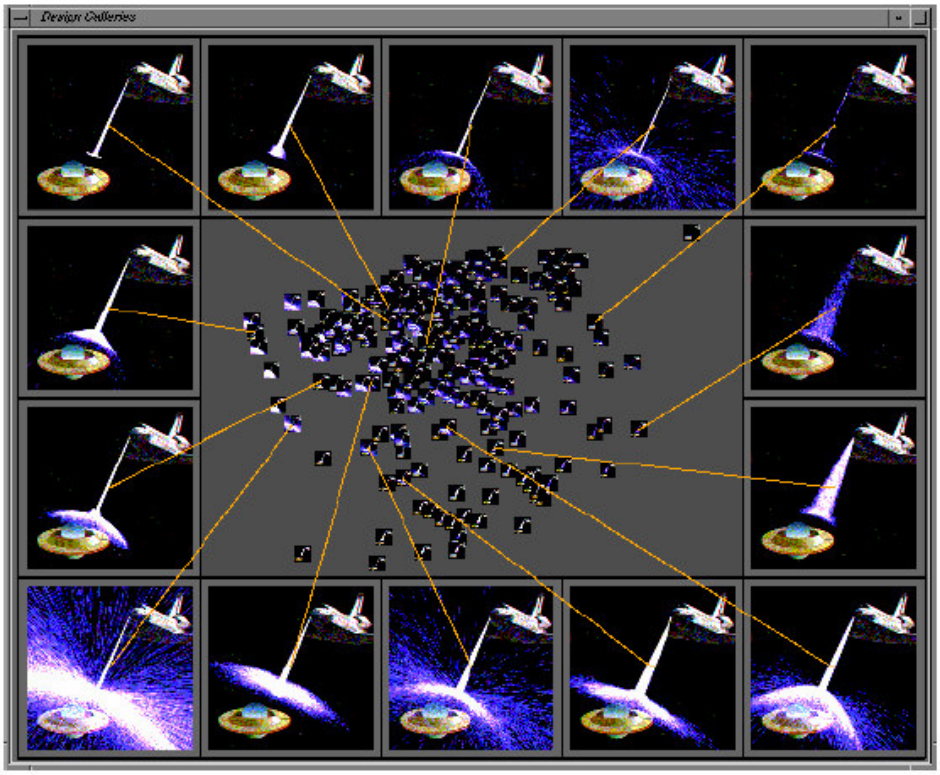


Figure 56. Représentation par une méthode MDS de l'ensemble des solutions pour la sélection des paramètres d'un système de particules (Image issue de [MAR97]).

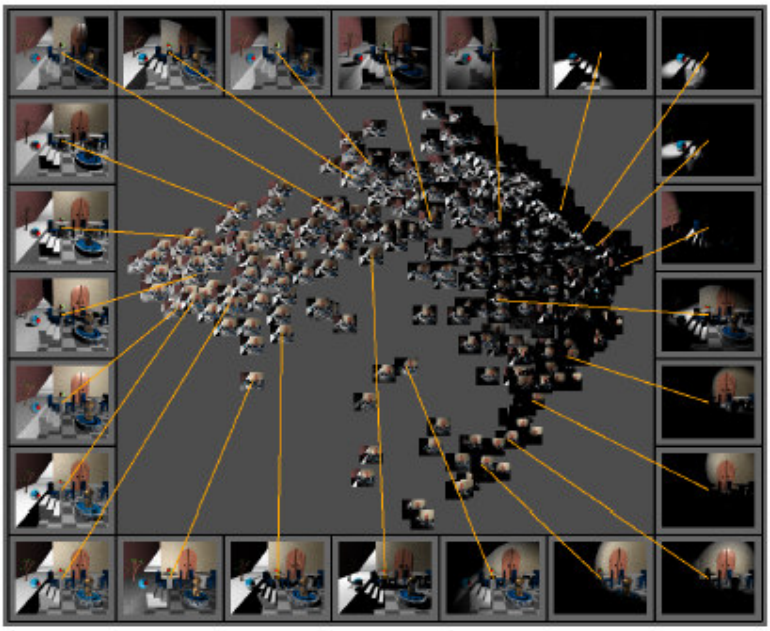


Figure 57. Représentation par une méthode MDS de l'ensemble des solutions pour un problème d'éclairage (Image issue de [MAR97]).

Comme le montrent les figures 58, 59 et 60, le fait de procéder à une projection sur un espace de dimension 3 (par la technique MDS) permet de conserver des informations qui deviennent imperceptibles quand on se limite à un espace de dimension 2. Par ailleurs, le fait de positionner les solutions dans l'espace au lieu d'un plan permet de limiter l'effet de *cluterring* observé en 2D.

La figure 58 présente un système de *Design Gallery* en 2D.

La figure 59 correspond à l'application de la méthode MDS en 3D.

La figure 60 est le résultat obtenu après une rotation des éléments de la figure 59. Une organisation particulière des données apparaît alors, organisation qui n'était pas révélée par la figure 59.

5.2.3 Les différentes étapes de la méthode proposée pour l'éclairage inverse.

Nous allons à présent reprendre les diverses étapes correspondant au schéma de fonctionnement que nous avons introduit au chapitre 4, mais en tenant compte des conclusions auxquelles nous avons abouties après avoir exploré l'approche à base de CSP employée fréquemment en modélisation déclarative.

Nous proposons donc une nouvelle approche des problèmes d'éclairage inverse, méthode que nous avons scindée en 6 étapes.

1. L'utilisateur indique les carreaux (ou les faces) qu'il souhaite éclairer. Ceci se fait de façon déclarative (Il y a une association entre des ensembles de carreaux de la scène et des éléments sémantiques de celle-ci).
2. L'utilisateur indique éventuellement les faces de la scènes pouvant contenir des sources lumineuses (Il s'agit donc d'une contrainte sur la position des sources de lumière).

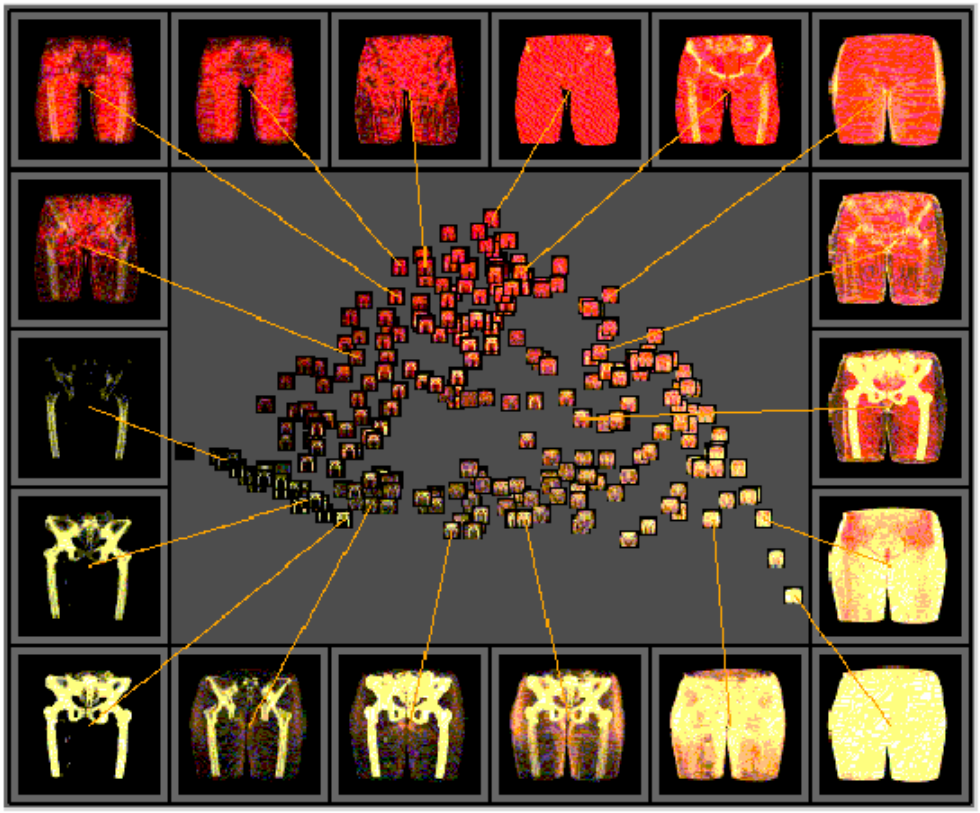


Figure 58. Un système de *Design Gallery* avec une représentation MDS en 2D (Image issue de [AND97]).

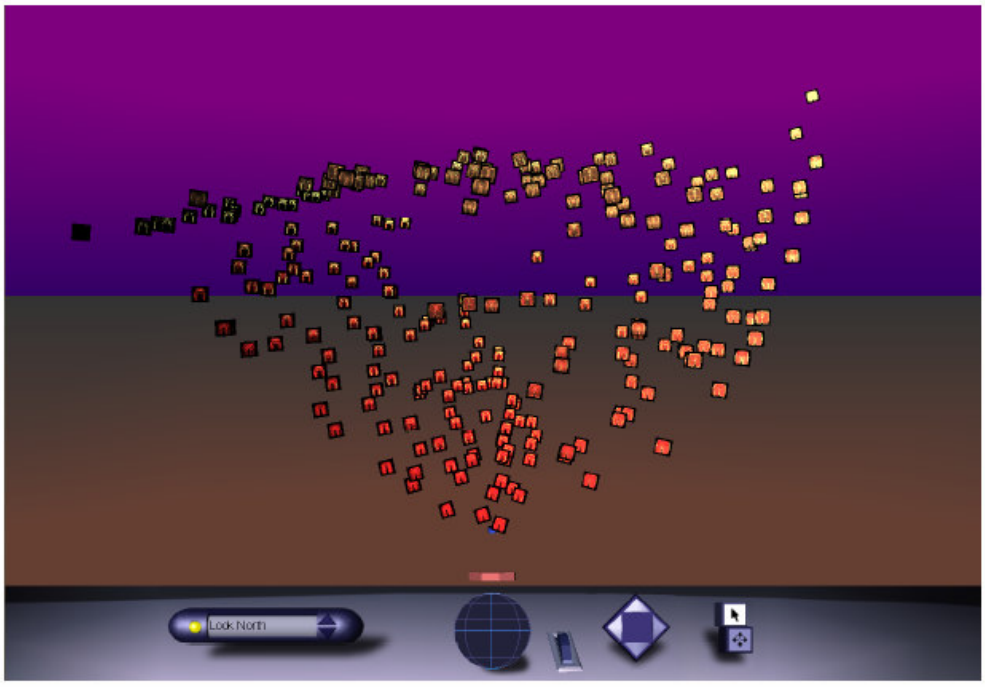


Figure 59. Le même système de *Design Gallery* avec une représentation MDS en 3D(Image issue de [AND97]).

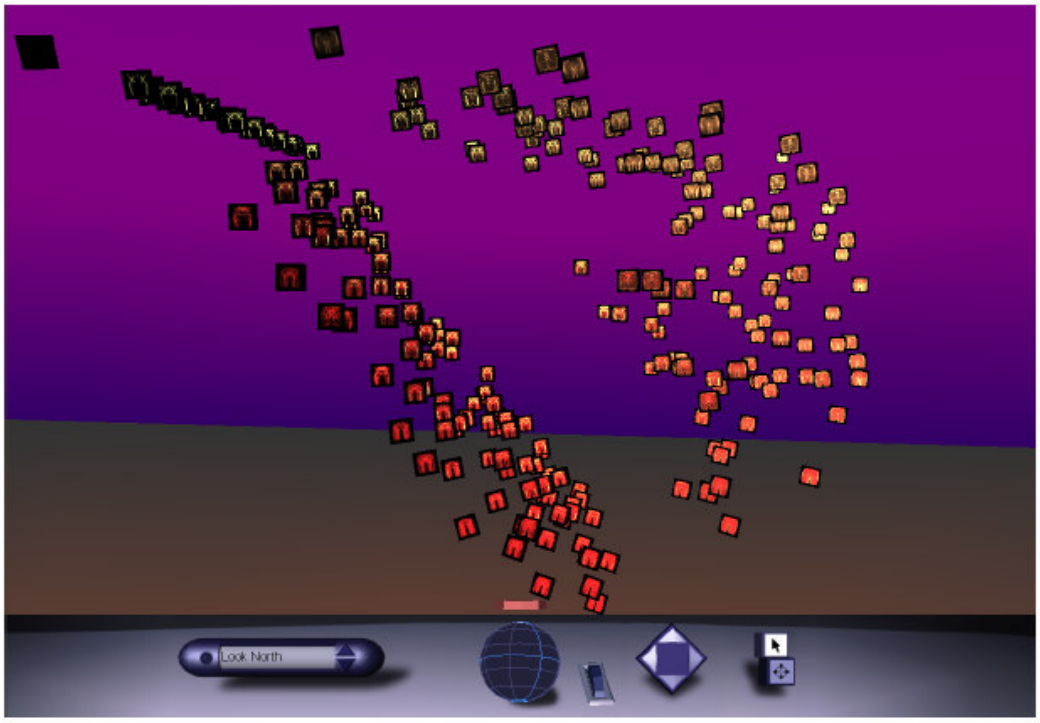


Figure 60. Visualisation de l'ensemble des solutions de la figure 12 après une rotation (Image issue de [AND97]).

---

3. Pour un éclairage direct, on sélectionne les carreaux des faces pouvant contenir des sources de lumière (avec un *raycasting* suivant la méthode de Monte-Carlo). Dans le cas de l'illumination globale, on *choisit* de garder toutes les faces potentiellement porteuses d'une source lumineuse.
4. En utilisant la modélisation des propriétés à base de sous-ensembles flous, on obtient un intervalle flou correspondant à la description de l'éclairage. Grâce à une α-coupe, on se ramène à un intervalle (bornes minimum et maximum de la radiosité désirée pour un carreau à éclairer).
5. La phase de dispersion (en suivant le vocabulaire de [MAR97]) :
   On essaie d'obtenir un ensemble de solutions assez différentes les unes des autres, c'est-à-dire un ensemble « bien équilibré ».
   Pour cela, on remaille les faces (ou parties de faces) que l'on a obtenues à l'étape 3. Les faces sont divisées en carreaux carrés appelés carreaux élémentaires. Une source lumineuse est une combinaison de ces carreaux élémentaires et dont la forme est rectangulaire. L'émittance attribuée aux carreaux élémentaires de la source lumineuse est fonction de l'intervalle déterminé à l'étape 4 et des facteurs de forme entre les carreaux émetteurs et les carreaux à éclairer.
6. La phase de présentation des scènes solutions (appelée phase d'arrangement des solutions dans [MAR97]) :
   On construit pour cela un graphe des solutions. Les premiers nœuds introduits dans le graphe forment une structure arborescente. Cette structure d'arbre permet une meilleure répartition des solutions et facilite la navigation de l'utilisateur dans l'ensemble des solutions (contrairement aux deux méthodes proposées par

[MAR97] qui produisent des représentations de l'ensemble des solutions incohérentes ou difficiles à appréhender par le designer).

Nous allons approfondir dans les paragraphes suivants les étapes 5 et 6 (puisque nous avons déjà détaillé les étapes précédentes dans notre étude).

5.2.4 Remaillage d'une partie d'une face de la scène.

On associe à la face étudiée un repère orthonormé (xOy), l'origine O correspondant au point du bas gauche de la face.

Pour des raisons de commodité d'expression, on va considérer que ce sont les carreaux les plus élevés de la face qui ont été atteints par les rayons cherchant à déterminer la visibilité depuis les carreaux à éclairer.

Appelons cet ensemble de carreaux potentiellement éclairants L.

Soit min_y le minimum des coordonnées en y des carreaux triangulaires ou rectangulaires de L.

Soit (D) la droite d'équation : y = min_y dans le repère (xOy).

On remaillage en introduisant des carreaux carrés sur la zone $\{(x,y) / y \geq min\_y\}$. On doit ainsi traiter de façon particulière les carreaux dont les côtés coupent la droite (D).

- Traitement des triangles dont l'intersection avec la droite (D) est non vide.
  On a deux cas possibles :
  - 1er cas : Il y a 2 points du triangle situés au-dessous de la droite (D) (cf. la figure 61).

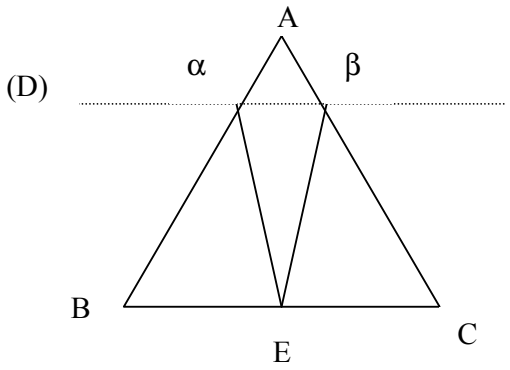


Figure 61. Cas où deux points du triangle sont positionnés au-dessous de la droite (D).

On a :

$$\alpha = [AB] \cap (D)$$
$$\beta = [AC] \cap (D)$$
E = milieu de [BC]

On enlève le triangle (ABC) et on insère les nouveaux carreaux triangulaires (αBE), (αβE) et (EβC).

  - 2ème cas : Il y a un seul point au-dessous de la droite (D) (cf. la figure 62).

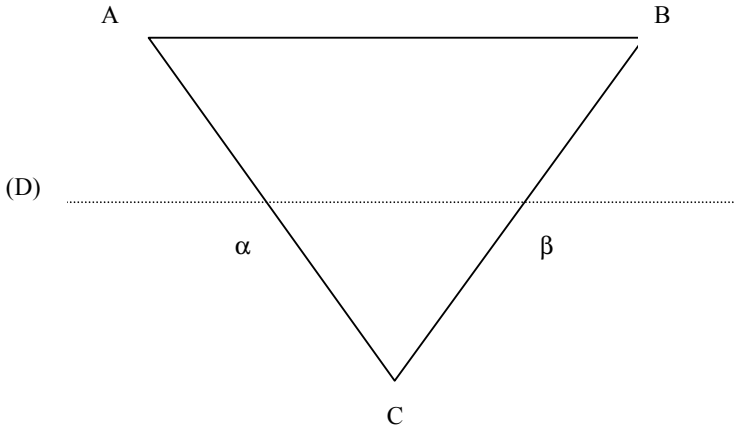


Figure 62. Cas où il n'y a qu'un point du triangle placé au-dessous de la droite (D).

On a :

$$\alpha = [AC] \cap (D)$$
$$\beta = [BC] \cap (D)$$

On supprime le carreau (ABC) de la liste des carreaux de la scène et on ajoute le triangle ($\alpha\beta$C) à celle-ci.

- Traitement des carreaux rectangulaires.
  On a aussi deux cas :
  - 1er cas : Pas de difficulté pour construire un nouveau carreau (L'un des côtés du rectangle est parallèle à (D)). Ce cas est illustré sur la figure 63.

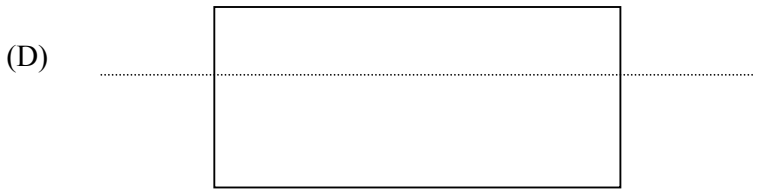


Figure 63. Cas où l'un des côtés du rectangle est parallèle à la droite (D)

  - 2ème cas : On décompose le rectangle en deux triangles et on se ramène à l'étude précédente (cf. la figure 64).

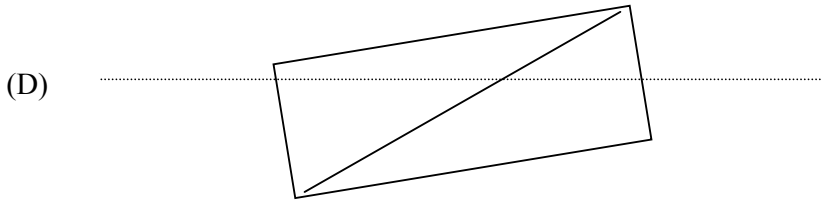


Figure 64. Décomposition d'un carreau rectangulaire en deux triangles pour se ramener à l'étude précédente.

5.2.5 Présentation des scènes solutions.

5.2.5.1 Construction du graphe.

Le graphe que l'on construit est fortement structuré (Tandis que [MAR97] construit un arbre à partir d'une partition d'un graphe complet, partition qui n'a pas vraiment de signification visuelle associée – ce qui fait que l'on a parfois l'impression que des solutions sont mal réparties dans l'arbre). Ceci est possible grâce aux nombreux paramètres que l'on peut manipuler pour construire un éclairage ([MAR97] ne dispose d'aucune information autre qu'un critère de distance entre deux solutions).

Dans les premiers niveaux du graphe, celui-ci apparaît comme une arborescence. Chaque arbre correspond à un nombre de sources lumineuses présentes dans la scène. Pour simplifier, nous allons nous limiter à une source lumineuse dans les explications qui suivent.

Nous allons décrire les différents niveaux du graphe qui est produit par l'algorithme.

Niveau 1 : Une solution ayant des valeurs moyennes pour les divers critères employés dans la fabrication du graphe. On peut voir ce nœud comme la racine de la structure arborescente du début du graphe.

Niveau 2 : On a k nœuds qui correspondent aux k faces (ou portions de faces) pouvant contenir une source lumineuse. Ce niveau n'existe pas si l'on a qu'une seule face dans laquelle on peut situer une lumière servant à satisfaire les objectifs d'éclairage.

Niveau 3 : On trouve 5 nœuds fils pour chacun des nœuds du niveau précédent. Chacun de ces nœuds correspond à une localisation dans la face du nœud père. Les 5 zones sont : haut gauche, haut droit, bas gauche, bas droit et centre.

Niveau 4 : On fait intervenir la taille de la source lumineuse à ce niveau. On choisit par convention une puissance de 2 pour l'aire. Si la face que l'on a choisie à ce niveau a pour dimensions n x m, un nœud du niveau précédent aura k fils au niveau 4, avec :

$$2^k \leq \max\left(n^2, m^2\right) < 2^{k+1}$$

Niveau 5 : La distance entre en compte à partir de ce niveau. Un nœud père du niveau 4 a x fils qui les sont les x scènes les plus proches de la scène correspondant au nœud père. *A ce niveau, la structure arborescente devient un graphe.*

On a vu qu'une notion de distance entre deux scènes intervient dans la construction du graphe. Explicitons l'heuristique nous avons introduite pour celle-ci.

Soit s une scène.

Soit $r(s) = \sum_{i=1}^{n} B_i A_i$ *où n est le nombre de patches de la scène.*

De façon intuitive, r(s) correspond à l'impact lumineux de la source de lumière dans la scène s.

Soient s_1 et s_2 deux scènes.

La distance $d(s_1,s_2)$ est alors définie par la formule suivante :

$$d(s_1, s_2) = \left| r(s_1) - r(s_2) \right|$$

En résumé, les critères intervenant successivement dans la construction du graphe sont donc :

- le nombre de sources de lumière,
- la position de la source de lumière,
- la taille de la source lumineuse,
- la distance entre les solutions obtenues.

5.2.5.2 Représentation du graphe.

La visualisation de l'information [HMM00] est un ensemble de techniques permettant de représenter des données structurées. Le fait que l'on dispose d'informations sur les données distingue la visualisation de l'information de méthodes employées en statistique descriptive où l'on cherche justement à découvrir des relations entre les données (à l'aide par exemple de l'analyse en composantes principales). Le but de la visualisation de l'information est de représenter de façon cohérente et claire un nombre important de données afin qu'une personne puisse prendre conscience des informations structurelles présentes dans ces données. Pour cela, il faut tenir compte du fait que l'utilisateur peut être amené à manipuler la représentation qu'on lui offre (ce qui implique une visualisation et des interfaces adaptées).

Deux problèmes apparaissent alors :

- Le contexte : on doit toujours avoir une vue globale (même imprécise) de la totalité des données.
- Le focus : on doit pouvoir sélectionner une donnée particulière.

A titre d'exemple :

Dans l'explorateur de Windows, on a le focus (puisque l'on peut arriver à sélectionner n'importe quel fichier d'une arborescence d'un disque). Par contre, on perd souvent le contexte (à force de développer la hiérarchie d'un disque, on ne voit plus la totalité des répertoires situés à la racine de celui-ci).

Une structure de graphe se rencontrant fréquemment (structure d'un site Web par exemple), un intérêt particulier a été porté à la représentation de données organisées sous forme de graphe.

L'un des algorithmes les plus étudiés et employés est celui de *spring embedding* (appelé aussi *force-directed method*). Chaque nœud du graphe est considéré comme une masse et chaque arête comme un ressort reliant 2 masses (nœuds). Des forces de répulsion ont tendance à faire s'éloigner les nœuds les uns des autres, tandis que des forces d'attraction rapprochent les nœuds reliés par une arête. Au départ, on positionne arbitrairement les nœuds dans un espace 2D ou 3D. Ensuite, l'algorithme déplace les nœuds en fonction des forces qui s'appliquent sur eux. L'algorithme prend fin quand on atteint un état d'équilibre.

Remarque : La vitesse de convergence vers l'équilibre, le fait que les nœuds ne se recouvrent pas et que les arêtes ne se coupent pas, sont les critères qui permettent le plus souvent de mesurer la qualité des différents algorithmes de *spring embedding*.

Un algorithme de ce type est implémenté dans le logiciel libre VGJ (*Visualizing Graphs with Java*). VGJ peut traiter en entrée des fichiers au format GML (*Graph Modelling Language* [HIM97]) et afficher des nœuds sous forme d'icônes. De plus, GML présente l'avantage de recourir à des fichiers textes dont la syntaxe est parente de celle de VRML. Ces fichiers GML sont ainsi facilement productibles par un langage de script.

A partir des critères de construction du graphe dont nous avons parlés précédemment, il nous suffit donc d'engendrer des fichiers GML, puis de visualiser les graphes avec VGJ.

5.2.6 Résultats.

5.2.6.1 Construction des scènes.

L'utilisateur fixe comme contrainte de position pour la (ou les) source(s) lumineuse(s) le fait de se trouver au plafond. Il veut éclairer le sommet de la boîte gauche de la scène. Il ne désire qu'une seule source lumineuse.

Le programme engendre toutes les sources lumineuses rectangulaires possibles à partir du nouveau maillage et en tenant compte de bornes sur la largeur et la longueur de la source

lumineuse (Il ne paraît pas raisonnable d'avoir une source lumineuse occupant tout le plafond).

Quelques chiffres :

Taille en x du plafond : 10.0
Taille en y du plafond : 10.0

Discrétisation choisie pour le remaillage du plafond :
Taille en x d'un carreau élémentaire du plafond : 1.0
Taille en y d'un carreau élémentaire du plafond : 1.0

Bornes pour la taille de la source lumineuse :
$1.0 \leq$ taille en x de la source < 5.0
$1.0 \leq$ taille en y de la source < 5.0

On obtient alors 1156 configurations possibles pour la source lumineuse (configurations qui correspondent donc à autant de scènes). On a 100 scènes dont la source de lumière occupe un seul carreau élémentaire.

Remarque : Avec l'algorithme choisi, la taille de la source lumineuse croît avec le numéro de la scène produite.

Les figures 65, 66 et 67 montrent trois exemples de scènes calculées.
Sur la figure 65 apparaît la première scène obtenue avec notre algorithme.
Sur la figure 66 est représentée la $100^{ème}$ scène obtenue, qui est aussi la dernière scène où la source de lumière se limite en taille à un carreau élémentaire.
Pour terminer cette présentation d'exemples, la figure 67 montre l'avant-dernière scène calculée.

Remarque sur la discrétisation pour le remaillage :
Des tests ont été effectués avec une source lumineuse de 0.2 x 0.2 (25 fois plus petite en terme d'aire). La source de lumière apparaît alors trop petite quand elle est constituée d'un seul carreau élémentaire, quoi que l'éclairage de la scène soit satisfaisant quand elle n'est pas placée sur un bord (conséquence de la nature diffuse des surfaces).

Temps de calcul :
Pour la construction des 1156 scènes et l'application d'une étape de la méthode de Feda-Purgathofer à chacune de ces scènes, on a un temps d'exécution de l'ordre de **3mn 30s** avec la même configuration machine que celle du 3.4.2.5.
La majorité du temps de calcul est prise par la méthode de Feda-Purgathofer. La détermination des positions possibles pour la source lumineuse est rapide puisque que l'on engendre automatiquement les configurations sur des faces et non pas à partir d'un ensemble L potentiellement complexe comme celui intervenant au chapitre 4.2.

5.2.6.2 Construction des sous-graphes.

En suivant la méthode de construction du graphe exposée au 5.2.5.1, on obtient comme premier sous-graphe celui de la figure 68.

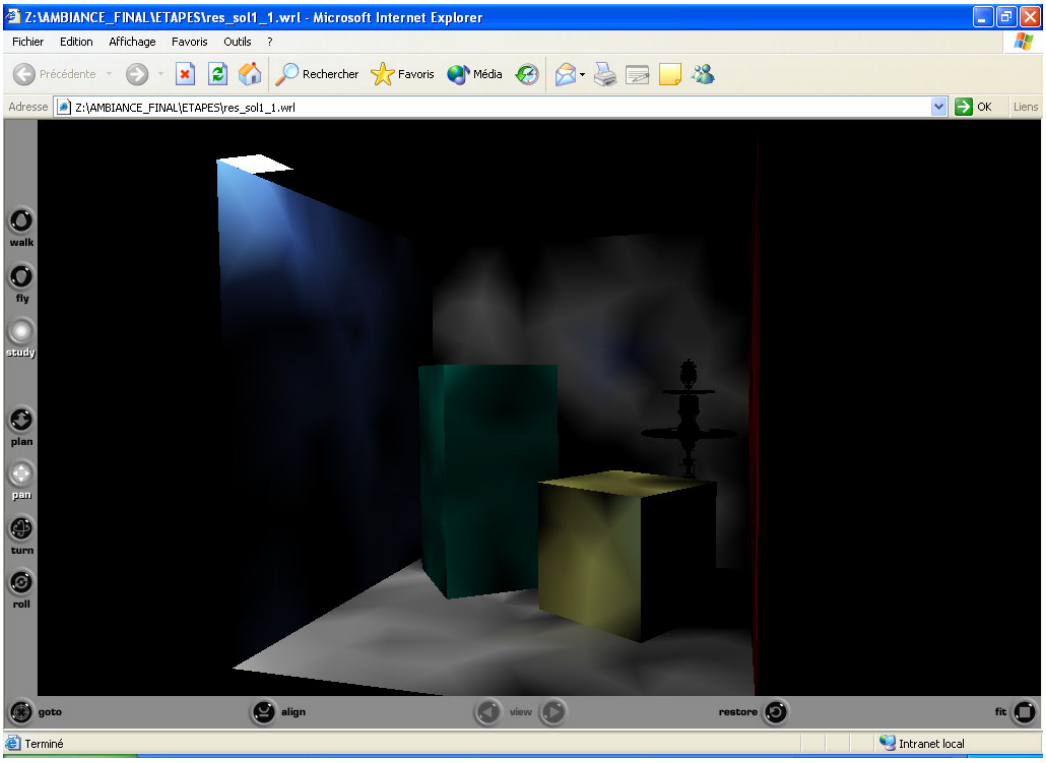


Figure 65. Première scène obtenue avec l'algorithme.

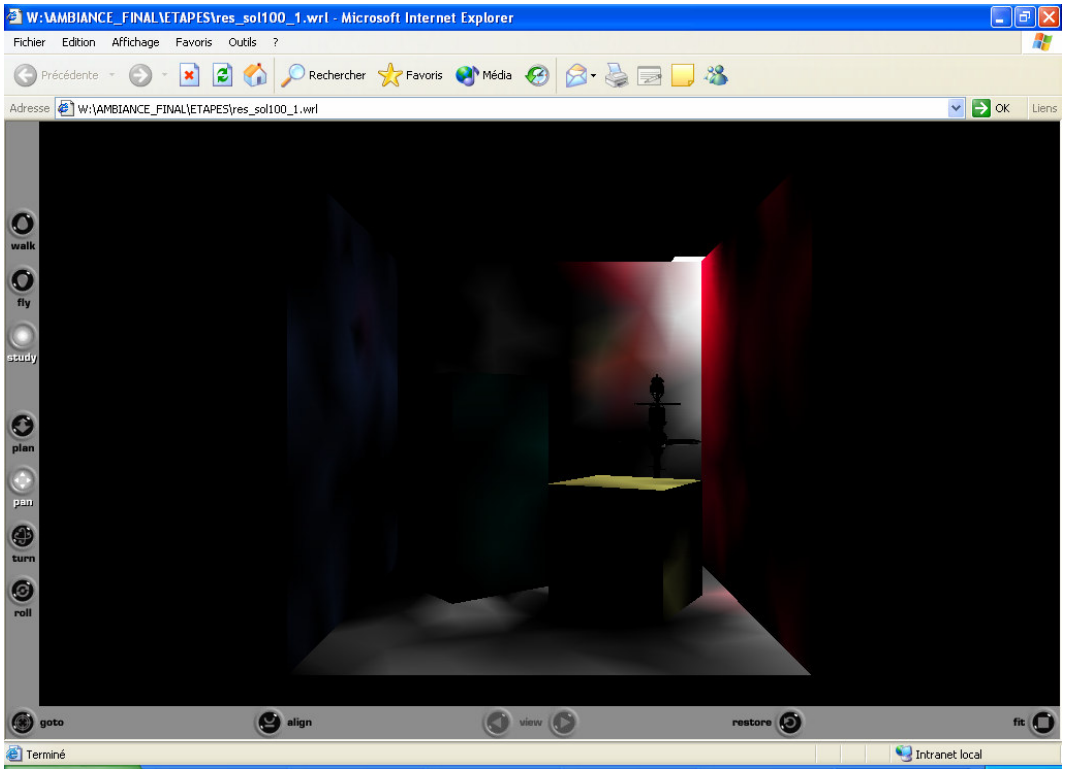


Figure 66. Centième scène calculée (ce qui correspond à la dernière scène engendrée où la source lumineuse est limitée à un carreau élémentaire).

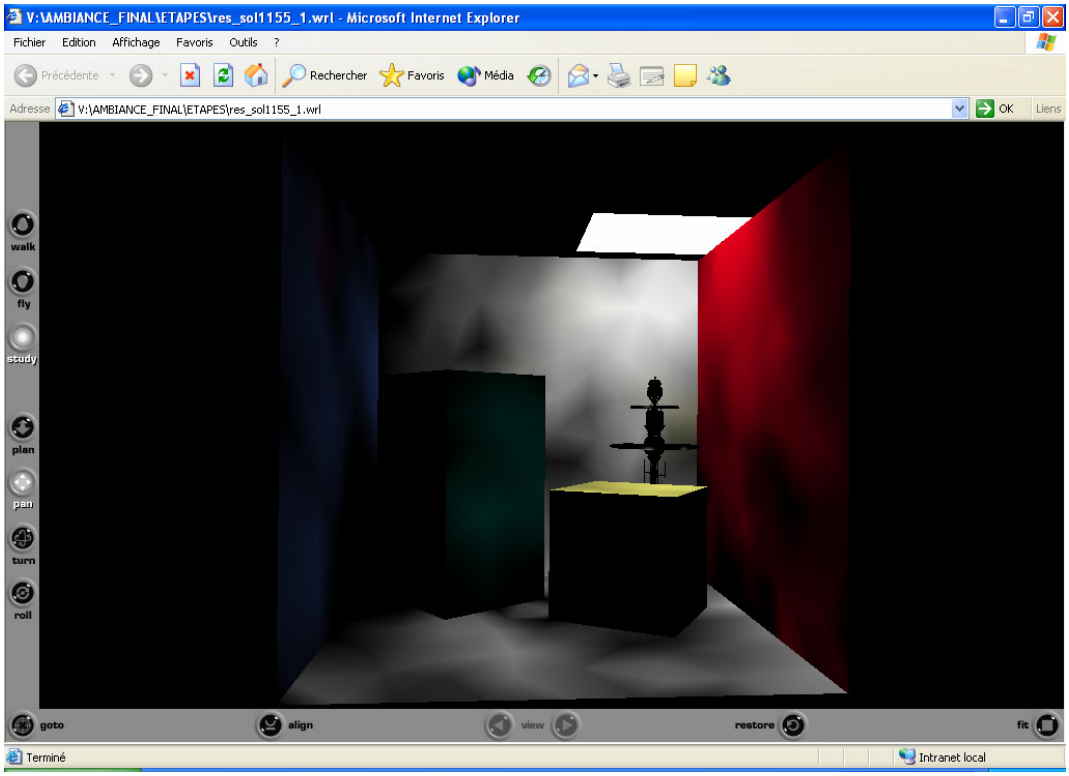


Figure 67. Avant-dernière scène produite.

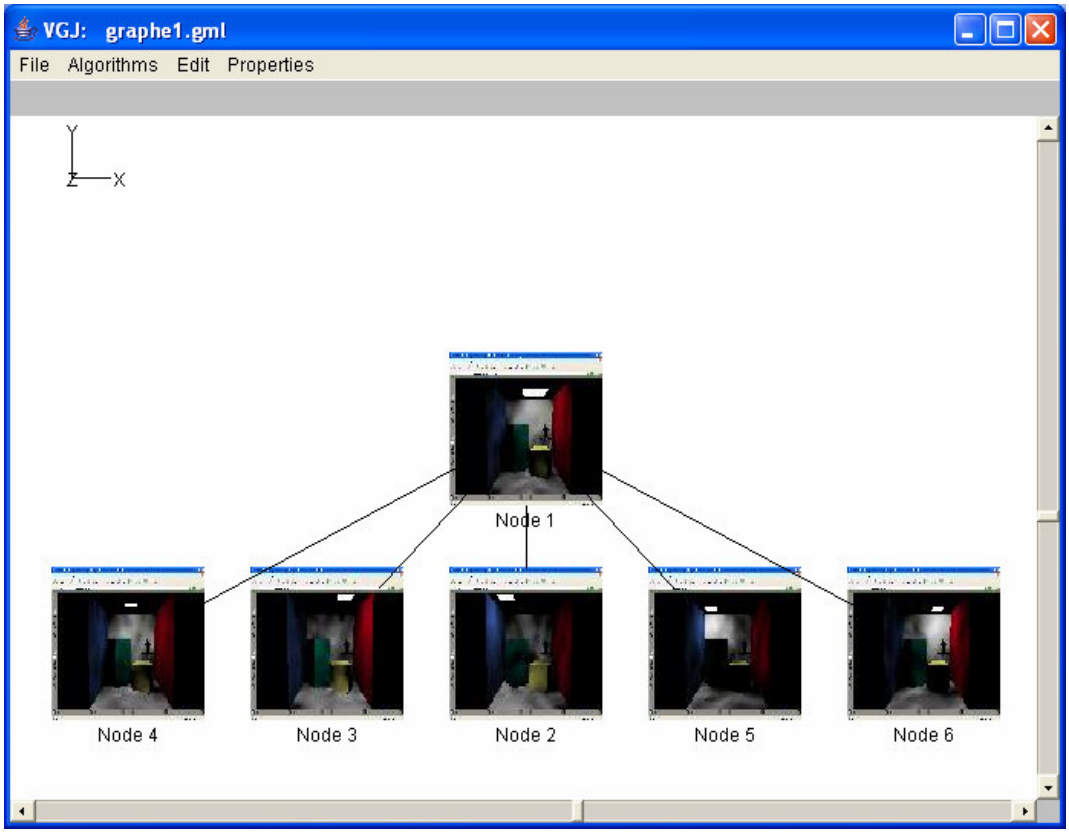


Figure 68. Premier sous-graphe du graphe des solutions présenté à l'utilisateur.

Si l'utilisateur sélectionne le nœud numéro 2 du graphe de la figure 68, il obtient le sous-graphe de la figure 69 :

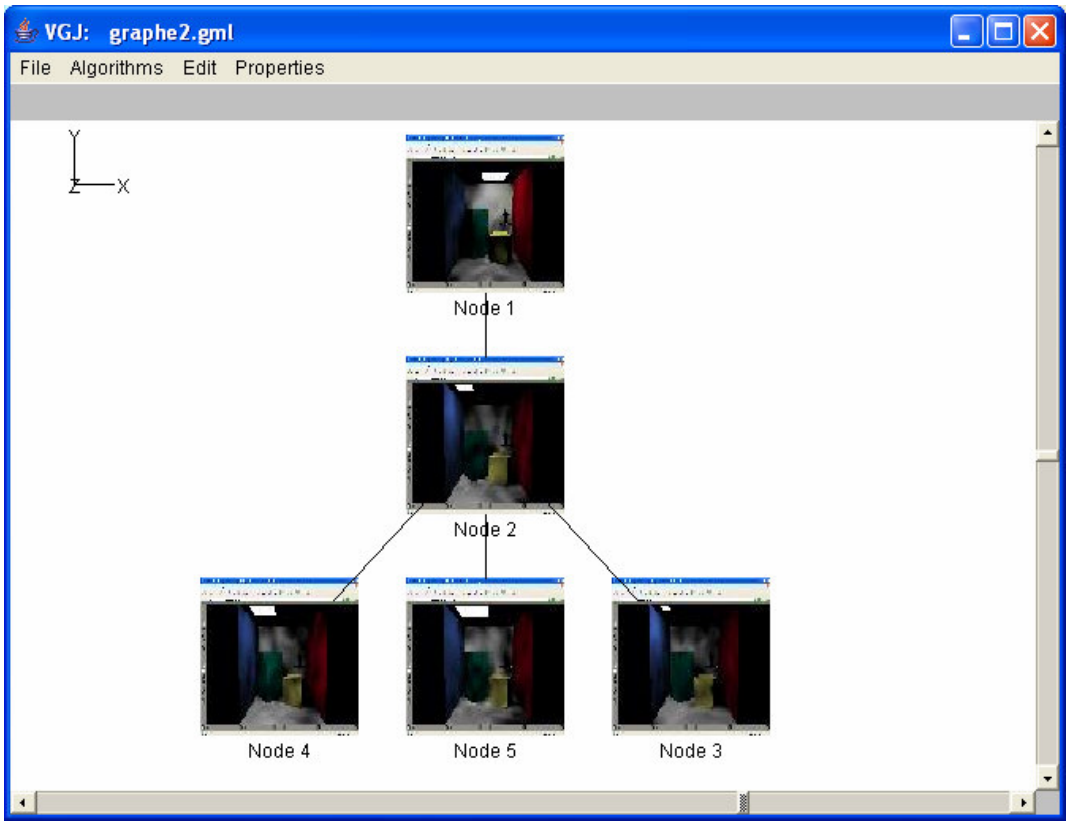


Figure 69. Sous-graphe présenté à l'utilisateur quand il choisit de voir les solutions en rapport avec le nœud 2.

Remarque : On a choisi une représentation sous forme d'arbre puisque dans ses premiers niveaux le graphe a une structure d'arbre. Cette représentation est plus intuitive que celle obtenue avec l'algorithme de *spring embedding* (qui est adapté à des graphes non structuré et de taille relativement importante).

5.2.7 Analyse des résultats obtenus.

Notre méthode se positionne comme une sorte d'intermédiaire entre l'éclairage inverse et les *Design Galleries*.

Par rapport aux techniques d'éclairage inverse, elle présente les avantages suivants :

- L'utilisateur peut utiliser une description de haut niveau pour spécifier ses objectifs d'éclairage. Il n'a pas à déterminer des paramètres dont l'influence sur le système est assez complexe à évaluer pour obtenir ce qu'il désire.
- Du fait d'aspect flou et imprécis d'une spécification déclarative de l'éclairage, plusieurs solutions se satisfaisant cette spécification sont produites, dont certaines que le concepteur va juger intéressantes alors qu'il ne pensait pas à celles-ci au moment où il formulait ses exigences.
- Les temps de calcul sont peu élevés en comparaison des méthodes d'optimisation coûteuses employées couramment dans les problèmes d'éclairage inverse.

Par rapport au paradigme des *Design Galleries*, elle possède les avantages suivants :

- L'espace de recherche des solutions est réduit grâce à une spécification déclarative des objectifs d'éclairage. Ceci permet d'éviter d'engendrer des scènes dont l'éclairage n'intéresse absolument pas l'utilisateur.
- Du fait de la réduction de l'espace de recherche, les temps de calcul pour la génération des scènes solutions sont beaucoup moins élevés.
- La présentation des solutions est de meilleure qualité car s'appuyant fortement sur la structure des vecteurs d'entrée du système.

On peut ainsi concevoir la méthode que nous avons présentée comme un compromis entre l'éclairage inverse et les *Design Galleries*.

Des problèmes demeurent cependant.

On peut aisément constater que les solutions calculées sont assez semblables d'un simple point de vue visuel. Ce sont avant tout la taille et la position de la source lumineuse qui sont prépondérantes pour distinguer deux solutions.

La similarité des solutions provient de l'utilisation de la radiosité comme modèle d'éclairement. Avec des sources lumineuses diffusant leur énergie dans toutes les directions, l'éclairage d'une pièce est assez semblable quand la position et la taille d'un carreau éclairant évoluent légèrement.

Par ailleurs, l'affichage du graphe nécessite une taille d'images assez petite, ce qui fait qu'elles sont assez peu lisibles.

5.2.8 Améliorations envisagées.

VGJ ne permet d'afficher que des graphes statiques. Il est bien évident que les choix des scènes par l'utilisateur se font de façon dynamique. Pour gérer ceci, il faut recourir à un algorithme qui permette d'afficher un graphe que l'on modifie progressivement. Un algorithme de *spring embedding* adapté a été proposé par [HEW98a] et utilisé dans [HEW98b] pour la visualisation du trajet suivi par un utilisateur surfant sur le Web.

Pour la taille trop petite des images représentant les scènes solutions, on pourrait améliorer l'interface utilisateur en introduisant une zone contenant un agrandissement de la solution actuellement sélectionnée par l'utilisateur.

La pertinence des images présentées à l'utilisateur dépend bien entendu du point de vue choisi pour leur production. Il est important de faire apparaître les sources lumineuses mais aussi les objets éclairés intervenant dans la description déclarative de l'éclairage. Or, il est généralement difficile de trouver un point de vue où ces deux parties de la scène puissent apparaître conjointement dans une image. Considérant que l'algorithme d'éclairage inverse assure l'illumination voulue par le concepteur pour certains objets de la scène, nous pensons donc qu'il est préférable de montrer en priorité les sources de lumière sur les images représentant la scène. Diverses heuristiques se rapportant au calcul d'un bon point de vue pour une scène ont été étudiées, notamment dans [BPD00]. Toutefois, ces techniques cherchent à donner la meilleure connaissance possible de la totalité d'une scène. Or, le fait que nous tenons à privilégier la visibilité des sources lumineuses constitue une contrainte à respecter. Il conviendrait ainsi d'essayer d'adapter les méthodes de [BPD00] en attribuant une importance élevée aux sources de lumière placées dans la scène.

Dans [AND97], nous avons vu que l'utilisation d'une représentation 3D permettait de percevoir des structures de l'ensemble solution qui n'apparaissaient pas dans une simple représentation 2D. Adapter un algorithme de *spring embedding* standard pour engendrer un graphe dans un espace 3D est trivial (Il suffit de travailler avec des équations portant sur des

vecteurs à 3 dimensions). On pourrait alors avoir des images solutions d'une taille plus grande – et donc plus lisibles. L'effet de *cluttering* est compensé par le fait que l'utilisateur peut aisément naviguer dans le monde virtuel dans lequel sont situées les solutions afin d'arriver à un point de vue où cet effet devient moins gênant.

Finalement, il serait intéressant de recourir à un autre modèle d'éclairement que la radiosité. Ceci nécessiterait une nouvelle définition de la distance entre deux scènes pour la construction du graphe des solutions.

Partie IV : Conclusion.

Les recherches présentées dans ce mémoire ont permis de développer une méthode pour spécifier de façon déclarative l'ambiance d'une scène.

La description de l'ambiance est donnée par le concepteur à l'aide de propriétés dont l'expression se fait avec un vocabulaire extrait d'une langue. La représentation interne des propriétés prend la forme d'intervalles flous. Cette modélisation des propriétés avait été employée par [DES98] dans le cadre de la modélisation déclarative. Nous l'avons étendue pour introduire des descriptions possédant un certain degré d'incertitude. Une première validation de la pertinence de notre approche a été accomplie pour la gestion de l'ambiance dans un paysage.

Nous avons par la suite étudié le problème de la création d'une ambiance pour une scène quelconque. Nous nous sommes concentrés sur l'une des composantes majeures de l'ambiance d'une scène : l'éclairage de cette scène. A partir d'une spécification déclarative de l'éclairage souhaité par le concepteur, nous avons proposé un algorithme permettant de trouver les caractéristiques des sources de lumière satisfaisant les désirs de l'utilisateur. Ces travaux en éclairage inverse, placés dans le cadre d'une spécification de nature déclarative des objectifs, nous ont conduit au problème de la gestion et de la représentation d'un ensemble très grand de scènes solutions correspondant à la spécification exigée. Nous avons alors amélioré des techniques issues du domaine de la visualisation de l'information grâce à la prise en compte des caractéristiques particulières de notre problème. Nous avons obtenu une classification plus judicieuse des diverses scènes solutions grâce à une meilleure répartition de celles-ci dans un graphe des solutions.

Le système que nous proposons permet un renforcement ou l'atténuation d'une ambiance dans une scène. En effet, nous ne modifions à aucun moment la géométrie de la scène. Or les divers éléments de la scène possèdent une ambiance intrinsèque qui va se composer avec l'ambiance issue d'autres éléments comme les sources lumineuses. Nous ne pouvons donc imposer une ambiance quelconque à une scène données.

Remarquons que nous n'avons pas donné de détails sur le traitement de toutes les propriétés concernant l'éclairage inverse que nous avons explicitées dans le chapitre 1.4. C'est par exemple le cas des sources de lumière en forme de disque, des spots ou des contraintes de placement dans une région (Par exemple, la contrainte : « La source lumineuse est située au centre du plafond »). Ces études complémentaires s'insèreraient a priori sans difficulté majeure dans notre schéma global de gestion de l'ambiance exprimée sous forme déclarative.

Au niveau de la manipulation des propriétés, nous pourrions rajouter des propriétés de comparaison ([DES98]). Elles prendraient par exemple la forme suivante : « La table est plus éclairée que la chaise ».

Les travaux futurs que nous envisageons concernent d'abord l'amélioration de l'interactivité avec l'utilisateur pour la phase d'exploration des solutions. Ceci passe par une adaptation à notre problématique d'un algorithme de *spring embedding* dynamique (avec disparition des images correspondant à des solutions que le concepteur a trouvées auparavant inintéressantes et apparition dans le graphe de nouvelles images directement liées à la scène actuellement sélectionnée par l'utilisateur). Cet algorithme de *spring embedding* pourrait être en 3D ce qui permettrait de représenter un plus grand nombre de solutions sans que se manifeste un effet de *cluttering*.

Nos recherches dans le cas de l'ambiance d'une scène quelconque ont été accomplies avec le modèle d'éclairement de la radiosité. Ayant étudié l'éclairage d'un lieu, le fait que la radiosité soit indépendante de la position de l'observateur correspondait bien à nos objectifs. Cependant, nous avons vu que c'était un modèle limité qui produisait des scènes visuellement très similaires. De plus, la représentation des scènes solutions s'effectuant avec des images (correspondant aux nœuds du graphe des solutions), on est bien contraint de fixer un point de vue pour la production de ces images qui vont orienter le concepteur lors de son exploration de l'ensemble des solutions. Bien que l'on puisse modifier l'expression du facteur de forme pour introduire des spots [CW93] et utiliser une méthode zonale pour obtenir des milieux participants [RT87], un autre modèle d'éclairement devrait autoriser la prise en compte de nouvelles propriétés pour une description plus fine de l'ambiance. Il faudrait alors adapter nos méthodes pour la génération de scènes solutions qui soient suffisamment différentes et pour la construction du graphe des solutions.

Références bibliographiques :

[AND97] B. Andalman et al.
Design Gallery Browsers Based on 2D and 3D Graph Drawing.
Symposium on Graph Drawing 97. Lecture Notes in Computer Science 1353. Springer-Verlag, pp. 322-329, 1997.

[ARV95] J. Arvo.
The Role of Functional Analysis in Global Illumination.
Proceedings of the 6th Eurographics Workshop on Rendering.

[ASBG02] J.-M. Alliot, T. Schiex, P. Brisset, F. Garcia.
Intelligence artificielle et Informatique théorique.
Cépadues. 2ème édition. 2002.

[BPD00] P. Barral, D. Plemenos, G. Dorme.
Scene understanding Techniques using a virtual Camera.
Short paper.EUROGRAPHICS'2000. August 20-25, 2000.

[CM97] M. Contensin, J.-L. Maltret.
Computer Aided Lighting for Architects and Designers.
IKM'97, Weimar, Février 1997.

[CON01] Contact, a VRML plug-in.
http://www.blaxxun.com/en/products/contact/index.html
2001 Blaxxun Technologies.

[CON02a] M. Contensin.
Redistributing Light.
Journal of WSCG : special Issue WSCG'2002. Pilsen, 4-8 February 2002.

[CON02b] M. Contensin.
Inverse Lighting Problem in Radiosity.
Journal of Inverse Problems in Engineering. Volume 10, Number 2. April 2002.

[CSF99a] A.C. Costa, A.A. Sousa, F.N. Ferreira.
Optimisation and lighting Design.
WSCG'99.

[CSF99b] A.C. Costa, A.A. Sousa, F.N. Ferreira.
Lighting Design : a Goal based Approach using Optimisation.
10th EG Workshop on Rendering, 1999.

[CW93] M.F. Cohen, J.R. Wallace.
Radiosity and realistic Image Synthesis.
Academic Press. 1993.

[DES95] Emmanuel Desmontils.
Formalisation des Propriétés en Modélisation déclarative à l'Aide des Ensembles flous.
Rapport de Recherche IRIN 106. Décembre 1995.

[DES98] Emmanuel Desmontils.
Le Projet CordiFormes : une Plate-forme pour la Construction de Modeleurs déclaratifs.
Thèse de Doctorat de la Faculté des Sciences et Techniques de Nantes. Janvier 1998.

[DES00] Emmanuel Desmontils.
Expressing Constraint Satisfaction Problems in Declarative Modeling using natural Language and fuzzy Sets.
Computer & Graphics 24 (2000). pp.555-568.

[DP97] Emmanuel Desmontils, Daniel Pacholczyk.
Interprétation du Vague dans la Description linguistique d'une Scène en Modélisation déclarative.
Rapport de Recherche IRIN 150. Avril 1997.

[ER97] J. Elorza, I.Rudomin.
An interactive System for solving inverse Illumination Problems using genetic Algorithms.
Computacion Visual 1997.

[GAC97] Louis Gacôgne.
Eléments de Logique floue.
Hermès. 1997.

[GAI03] V. Gaildrat.
Modélisation déclarative d'environnements virtuels : Création de scènes et formes complexes par l'énoncé de propriétés et l'emploi d'interactions gestuelles.
Habilitation à Diriger des Recherches. Université Paul Sabatier. Janvier 2003.

[GOL94] D.E. Goldberg.
Algorithmes génétiques.
Addison-Wesley, 1994 (Traduction française).

[GUM02] S. Gumhold.
Maximum Entropy Light Source Placement.
Proc. of the Visualization 2002 Conference, pp. 275-283. IEEE Computer Society Press, October 2002.

[HMM00] I. Herman, G. Melançon, M.S. Marshall.
Graph Visualization and Navigation in Information Visualization: a Survey.
IEEE Transactions on Visualization and Computer Graphics 2000.

[HEW98a] M.L. Huang, P. Eades, J. Wang.
Online Animated Graph Drawing using a Modified Spring Algorithm.
Proc. of the 21st Australasian Computer Science Conference, pp. 17-28. 1998.

[HEW98b] M.L. Huang, P. Eades, J. Wang.
WebOFDAV – Navigating and Visualizing the Web on-line with animated Context Swapping.
WWW7: 7th International World Wide Web Conference. 1998.

[HIM97] M. Himsolt.
GML: a portable Graph File Format.
Technical Report, Universität Passau, 94030 Passau, Germany, 1997.

[HMH95] V. Harutunian, J.C. Morales, J.R. Howell.
Radiation Exchange within an Enclosure of diffuse-gray Surfaces : The inverse Problem.
ASME/AIAA International Heat Transfer Conference, 1995.

[HSA91] P. Hanrahan, D. Salzman, L. Aupperle.
A rapid Hierarchical Radiosity Algorithm.
SIGGRAPH'91 Proceedings.

[IES00] Illuminating Engineering Society of North America (IESNA).
The IESNA Lighting Handbook: Reference and Application.
9th edition. Ed M.S. Rea. New York: IESNA. 2000.

[ING98] L. Ingber.
Adaptive Simulated Annealing.
ftp://ftp.ingber.com/pub
Lester Ingber Research ; 1993-1998.

[JPP02a] Vincent Jolivet, Dimitri Plemenos, Patrick Poulingeas.
Inverse Direct Lighting with a Monte Carlo Method and Declarative Modelling.
CGGM'2002, Amsterdam, April 2002.
Lecture Notes in Computer Sciences 2330, Springer.

[JPP02b] Vincent Jolivet, Dimitri Plemenos, Patrick Poulingeas.
Declarative Approach of inverse direct lighting Problems.
3IA'2002. Limoges, 14-15 mai 2002.

[JPP04] Vincent Jolivet, Dimitri Plemenos, Patrick Poulingeas.
Declarative Specification of Ambiance in VRML Landscapes.
CGGM'2004. Krakow, June 2004.
Lecture Notes in Computer Sciences 3039, Springer.

[KAH96] J. Kahrs.
Pixel Cinematography. Lighting for Computer Graphics.
Notes for Course #30, SIGGRAPH 96.

[KAJ86] J.T. Kajiya.
The rendering Equation.
Computer Graphics, 20(4), pp. 143-150, August 1986.

[KPC93] J.K. Kawai, J.S. Painter, M.F. Cohen.
Radioptimization – Goal Based Rendering.
SIGGRAPH'93.

[KUM92] V. Kumar.
Algorithms for Constraint Satisfaction Problems: A Survey.
AI Magazine 13, pp. 32-44. 1992.

[LD95] M. Lucas, E. Desmontils.
Les modeleurs déclaratifs.
Revue internationale de CFAO et d'informatique graphique, 1995. Vol. 10, N°6, pp. 559-585. Hermès (ISSN 0298-0924).

[LTG92] D. Lischinski, F. Tampieri, D. Greenberg.
Discontinuity Meshing for accurate Radiosity.
IEEE Computer Graphics & Applications, Vol. 12, N°6, pp. 25-39. 1992.

[MAR97] J. Marks et al.
Design Galleries: A General Approach to Setting Parameters for Computer Graphics and Animation.
SIGGRAPH 97.

[MG97] S. Marschner, D. Greenberg.
Inverse Lighting for Photography.
IS&T/SID Fifth Color Imaging Conference. November 1997.

[MOE01] M. Moeck .
On Top-Down Architectural Lighting Design.
CAAD Futures 2001, Kluwer Academic Publishers.
ISBN 0-7923-7023-6.

[MS01] I. Miguel, Q. Shen.
Solution Techniques for Constraint Satisfaction Problems: Foundations.
Artificial Intelligence Review 15. pp. 243-267. Kluwer Academic Publishers. 2001.

[OH95] M. Oguma, J.R. Howell.
Solution of the two-dimensional blackbody inverse radiation Problem by inverse Monte-Carlo Method.
ASME/JSME Thermal Engineering Conference, 1995.

[PF92] P. Poulin, A. Fournier.
Lights from Highlights and Shadows.
March 1992, Symposium on Interactive 3D Graphics.

[PLE95] D. Plemenos.
Declarative modeling by hierarchical decomposition.
The actual state of the MultiFormes project.
GraphiCon'95, St Petersbourg, 1-5 juillet 1995.

[PP01a] G. Patow, X. Pueyo.
A Survey on Inverse Reflector Design and Light Source Distribution Problems.
Institut d'Informàtica i Aplicacions, Universitat de Girona.
Communication privée.

[PP01b] G. Patow, X. Pueyo.
A Survey on Inverse Emittance and Inverse Reflectometry Computation Problems.
Institut d'Informàtica i Aplicacions, Universitat de Girona.
Communication privée.

[PP03] G. Patow, X. Pueyo.
A Survey on Inverse Rendering Problems.
Computer Graphics Forum. Volume 22 pp. 663-687. 2003.

[PRJ97] P. Poulin, K. Ratib, M. Jacques.
Sketching Shadows and Highlights to position Lights.
Proceedings of Computer Graphics International 1997.

[PTVF92] W. H. Press, S.A. Teutolsky, W.T. Vetterling, B.P. Flannery.
Numerical Recipes in C.
The Art of Scientific Computing. Second Edition.
Cambridge University Press. 1992.

[RT87] H.E. Rushmeier, E. Torrance.
The Zonal Method for Calculating Light Intensities in the Presence of a Participating Medium.
Computer Graphics 21, pp. 293-302. July 1987.

[RUT98] Z. Ruttkay.
Constraint Satisfaction – a Survey.
CWI Quaterly Vol. 11, pp. 123-161. 1998.

[SA03] Mark Segal, Kurt Akeley.
The OpenGL® Graphic System: A Specification (Version 1.5).
1992-2003.

[SDSAG93] C. Schoeneman, J. Dorsey, B. Smits, J. Arvo, D. Greenberg.
Painting with Light.
SIGGRAPH'93.

[SHI91] P. Shirley.
Radiosity via Ray Tracing.
Graphics Gems II, pp. 306-310, Academic Press, San Diego. 1991.

[SL01] R. Shacked, D. Lischinski.
Automatic Design using a Perceptual Quality Metric.
Computer Graphics Forum (Proceedings of Eurographics 2001). Volume 20, pages C-215-C-226.
September 2001.

[SIR96] Daniel Siret.
Sunlighting Design: an Inverse Approach of Simulation for CAD Tools. Advances in Computer-Aided-Design, CADEX'96. Hagenberg, September 1996.

[SIR97] Daniel Siret.
Propositions pour une Approche Déclarative des Ambiances dans le Projet Architectural. Application à l'Ensoleillement.
Thèse de Doctorat. Université de Nantes, June 1997.

[TEL92] P. Teller.
A contemporary Look at Emergence.
« Emergence or Reduction ? » Walter de Gruyter 1992, pp. 139-153.

[TER04] Terragen™.
http://www.planetside.co.uk/terragen/
1998-2004 Planetside Software.

[TSA93] E. Tsang.
Foundations of Constraint Satisfaction.
Computation of Cognitive Science. Academic Press. 1993.

[VRML97] VRML97 International Standard**.**
ISO/IEC 14772-1:1997.

[VS03] P.-P. Vazquez, M. Sbert.
Perception-Based Illumination Information Measurement and Light Source Placement.
ICCSA'2003.
Lecture Notes in Computer Science 2003. Volume 2669. pp. 306-316.

[WAR94] G.J. Ward.
The RADIANCE Lighting Simulation and Rendering System.
SIGGRAPH'94.

Spécification déclarative de l'ambiance d'une scène.

Résumé :
Les modeleurs couplés à des moteurs de rendu ont fait d'énormes progrès ces dernières années. Cependant, l'utilisateur de ceux-ci ne dispose pas d'outils lui permettant de spécifier des propriétés de haut niveau pour caractériser l'ambiance d'une scène.
A cette fin, nous avons étudié un modèle à base de sous-ensembles flous pour gérer de telles propriétés. Nous avons introduit dans celui-ci la possibilité de donner des spécifications incertaines. Le modèle a été testé avec l'ambiance d'un paysage.
L'éclairage étant une caractéristique fondamentale de l'ambiance, nous avons ensuite abordé le problème de l'éclairage inverse. Nous avons proposé pour celui-ci une nouvelle méthode basée sur une technique de Monte-Carlo dans le cadre de la radiosité. Nous avons adapté cette méthode pour qu'elle soit traitable par des CSP. Finalement, le problème du classement des nombreuses scènes solutions obtenues a été étudié, avec l'utilisation de techniques issues de la visualisation de l'information.

Mots-clés :
Modélisation déclarative,
Sous-ensembles flous,
Eclairage inverse,
Radiosité,
Méthode de Monte-Carlo,
Problèmes de satisfaction de contraintes,
Visualisation de l'information.

Declarative Specification of Ambiance in a Scene.

Abstract :
Modellers coupled to rendering tools made enormous progress these last years. However, the user of those does not have tools allowing him to specify high level properties in order to characterize the ambiance of a scene.
For this purpose, we studied a fuzzy subsets-based model to manage such properties. We introduced in this model the ability of giving uncertain specifications. The model has been tested with the ambiance of a landscape.
Lighting being a fundamental parameter of ambiance, we then tackled the problem of inverse lighting. We have proposed for this one a new method based on a Monte Carlo technique within the lighting model of radiosity. We adapted this method to be manageable by Constraint Satisfaction Problems. Finally, the problem of the classification of the founded scenes has been studied, with the use of techniques coming from information visualization.

Keywords :
Declarative Modelling,
Fuzzy subsets,
Inverse Lighting,
Radiosity,
Monte Carlo Methods,
Constraints Satisfaction Problems,
Information Visualization.