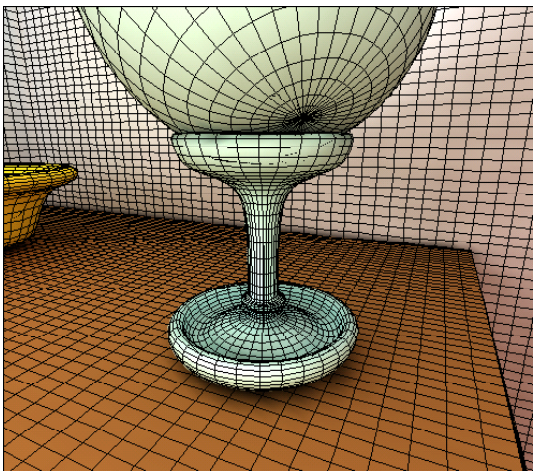
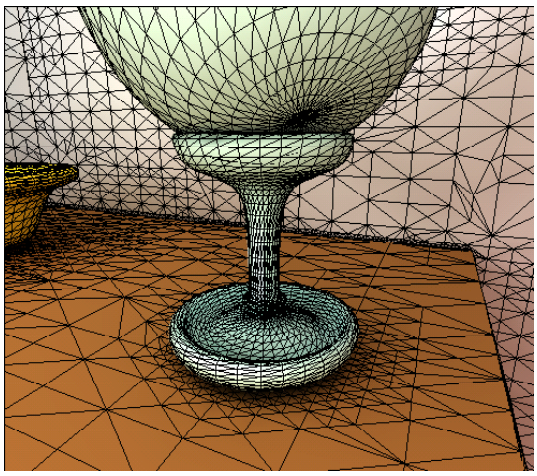



Stage MIM1 :

Radiosité discrète

David ROGER

Juin-Juillet 2002

Introduction

Ce stage s'est déroulé du 27 Mai au 10 juillet sous la direction de Rémy Malgouyres au *Laboratoire de Logique, Algorithmique et Informatique de Clermont1* (LLAIC1) situé dans les locaux de l'IUT d'informatique de Clermont-Ferrand.

L'équipe du LLAIC1 a mis au point une nouvelle technique de rendu de la luminosité pour la synthèse d'images : la radiosité discrète. Le stage consistait en l'étude et l'amélioration de cette technique.

La méthode de radiosité discrète est mise en oeuvre dans un logiciel créé par Rémy Malgouyres et amélioré simultanément à Clermont Ferrand et en Pologne par des étudiants.

Les objectifs étaient :

1. étude des techniques classiques de rendu
2. étude de la méthode de radiosité discrète
3. ajout d'une fonctionnalité au logiciel : l'affichage des facettes
4. développement et implémentations d'améliorations de la technique :
   - subdivision adaptative des facettes
   - suppression des points simples non visibles

Les techniques classiques de rendu sont étudiées au chapitre 1 et en annexe.
Le chapitre 2 décrit les bases théoriques et les méthodes de la radiosité discrète.
Les chapitres 3 et 4 traitent quant à eux de ce que j'ai réalisé pendant le stage.

Bonne lecture !

Table des matières

1 Techniques classiques de rendu **5**

1.1 Z-buffer 5

1.1.1 La scène en trois dimensions 5

1.1.2 Algorithme 6

1.1.3 Un modèle simple d'illumination 6

1.1.4 Lissage 8

1.1.5 Les faiblesses du Z-buffer 9

2 La radiosité discrète **11**

2.1 L'équation de base 11

2.2 Discrétisation de la scène 3D 12

2.2.1 Les structures de données 12

2.2.2 La n-connexité 13

2.2.3 Présentation du problème 13

2.2.4 Algorithme de discrétisation 14

2.3 Résolution de l'équation 14

2.3.1 Discrétisation de l'équation 14

2.3.2 Méthodes de Jacobi et de Gauss-Siedel 16

2.4 Affichage 17

2.5 Comparaison avec le lancer de rayon 18

3 Subdivision adaptative des facettes **20**

3.1 Triangulation des facettes 21

3.1.1 Généralités sur les facettes 21

3.1.2 Algorithme de triangulation 22

3.1.3 Détail de l'algorithme 23

3.2 Subdivision des facettes 25

3.2.1 Contraintes 25

3.2.2 Critère de division 26

3.2.3 Initialisation du graphe des sommets 30

3.3 Résultats expérimentaux 30

4 Elimination des points simples non visibles **33**
4.1 Les points simples 34
4.2 Détection des points simples 35
4.2.1 Voisinages géodésiques et nombre topologique 35
4.2.2 Caractérisation des points simples 35
4.3 Implémentation 35
4.3.1 Détection des points simples à l'aide de la caractérisation 35
4.3.2 Visibilité d'un point simple 36
4.4 Résultats expérimentaux 36

A Méthode du lancer de rayons **38**
A.1 Principe 38
A.2 Calcul des rayons 39
A.3 Les limites du lancer de rayons 39

B Méthode de la radiosité classique **41**

C Bibliographie **43**

Chapitre 1

Techniques classiques de rendu pour la synthèse d'image

Il est indispensable de connaître ces techniques car elles seront utilisées par la méthode de radiosité discrète, directement ou sous forme de variante.

Seule la technique du Z-buffer est décrite ici car elle est utilisée dans la méthode de radiosité. Les techniques du lancer de rayons et de la radiosité sont placées en annexe.

1.1 Z-buffer

Le Z-buffer est un algorithme d'élimination des parties cachées. Son principe est de parcourir tous les objets de la scène et de n'en afficher que les parties visibles.

1.1.1 La scène en trois dimensions

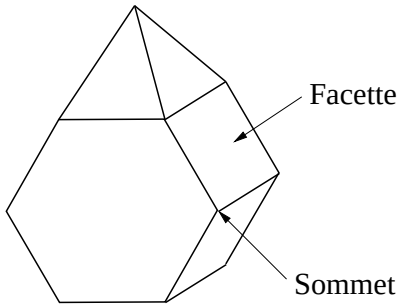


FIG. 1.1 – Un polyèdre

Pour pouvoir appliquer l'algorithme du Z-buffer, tous les objets de la scène doivent être des polyèdres. Ces polyèdres sont définis par la donnée de leurs sommets et de leurs facettes (voir fig.1.1).

L'affichage de la scène consiste à calculer la projection de la scène sur un plan (le plan de l'écran). Quitte à effectuer une rotation, on peut supposer que le plan de l'écran est le plan Oxy. On maintient à jour durant l'algorithme une matrice *minz* telle que :

$$\begin{aligned} minz[x][y] &= \min\{z_M / \text{ M point rencontré par l'algorithme} \\ &\quad \text{se projetant sur le pixel}(x,y)\} \end{aligned}$$

On a également une matrice de couleurs c telle que $c[x][y]$ est la couleur courante du pixel (x, y).

1.1.2 Algorithme

Initialiser la matrice minz à l'infini.
Initialiser la matrice c à la couleur de fond.
Pour chaque objet de la scène :
    Pour chaque facette de l'objet :
        Calculer la projection de la facette sur le plan de l'écran. On obtient ainsi un polygone P.
        Pour chaque point M=(x,y) de l'interieur de P :
            Calculer z(M) à l'aide de l'équation du plan de la facette.
            Si z(M) < minz[x][y] :
                Faire minz[x][y]=z(M).
                Calculer la couleur de M.
                Mettre à jour c[x][y].
Afficher l'image à l'aide de la matrice c.

1.1.3 Un modèle simple d'illumination

Pour calculer la couleur d'un point, il faut utiliser un modèle d'illumination. Celui qui est couramment utilisé avec l'algorithme du Z-buffer est très simple.
Dans ce modèle, la lumière émise par un point est la somme de trois termes :
- la lumière ambiante
- la réflection diffuse
- la réflection spéculaire

Lumière ambiante

La lumière ambiante émane de tous les objets, avec une même intensité dans toutes les directions. L'intensité de cette lumière en un point d'un objet est du type :

$$I_1 = k_a I_a$$

où I_a est l'intensité de la lumière ambiante de la scène et k_a est un coefficient compris entre 0 et 1 caractéristique de l'objet considéré.

Réflection diffuse

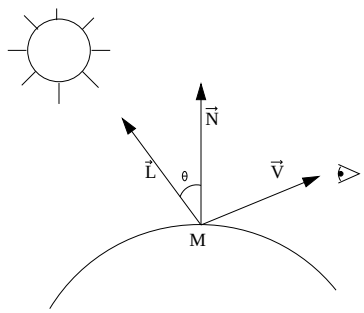


FIG. 1.2 – Vecteurs pour la réflection diffuse

La scène comporte une source lumineuse ponctuelle. On se place en un point M d'un objet et on définit le vecteur $\overrightarrow{N}$ normal à l'objet, le vecteur $\overrightarrow{V}$ dirigé vers l'observateur, le vecteur $\overrightarrow{L}$ dirigé vers la source lumineuse et l'angle θ comme sur la figure 1.2. Le terme de réflection diffuse vaut alors :

$$I_2 = \begin{cases} k_{rd} I_s \cos\theta & \text{si } \theta > 0 \\ 0 & \text{sinon} \end{cases}$$

où I_s est l'intensité de la source ponctuelle et k_{rd} est un coefficient compris entre 0 et 1 caractéristique de l'objet considéré.

Réflection spéculaire

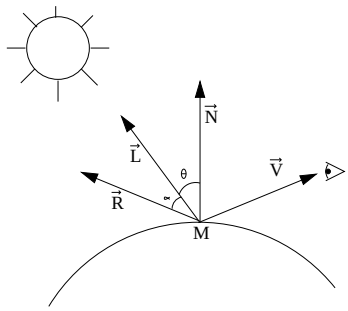


FIG. 1.3 – Vecteurs pour la réflection spéculaire

Cette réflection est celle que l'on peut observer sur des surfaces brillantes : la lumière est davantage réfléchie dans les directions voisines de la direction de réflection idéale de Descartes.
On définit le vecteur $\overrightarrow{R}$ comme le symétrique de $\overrightarrow{V}$ par rapport à $\overrightarrow{N}$ et l'angle α comme sur la figure 1.3. Le terme de réflection spéculaire est alors proportionnel à $\cos^{n_s}\alpha$ où n_s

est *l'exposant spéculaire.*
On a ainsi l'équation d'éclairement au point M :

$$\begin{aligned} I &= I_1 + I_2 + I_3 \\ &= k_a I_a + I_s(k_{rd} \cos\theta + k_s \cos^{n_s} \alpha) \end{aligned}$$

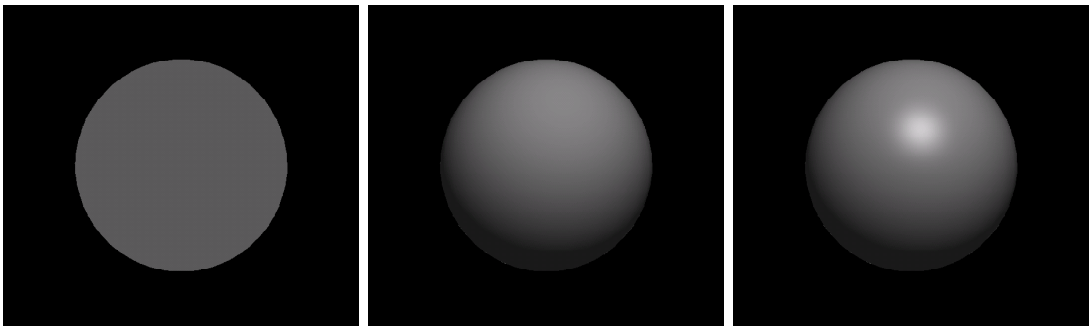


FIG. 1.4 – Eclairage ambiant, réflection diffuse et réflection spéculaire

Raffinements

On peut modéliser l'atténuation de la lumière avec la distance en multipliant les termes de réflection diffuse et spéculaire par un coefficient qui décroit en fonction de la distance entre le point M et la source lumineuse.

On peut modéliser des sources ponctuelles directionnelles (c'est à dire qui emettent leur lumière dans une direction privilégiée) en multipliant les termes de réflection diffuse et spéculaire par un coefficient du type $\cos^{n_d} \varphi$ où φ est l'angle entre la direction de la source lumineuse et $\overrightarrow{L}$, et n_d est un exposant superieur à 1.

1.1.4 Lissage

Dans le modèle d'illumination précédent, les arêtes des polyèdres apparaissent nettement. Ceci est gênant lorsque l'on souhaite représenter un objet arrondi (par exemple une sphère).
Pour palier à ce phénomène, on met en place des méthodes de lissage. Les deux principales sont le lissage de Gouraud et le lissage de Phong.

Lissage de Gouraud

Au lieu de calculer la couleur de chaque point d'une facette, on calcule uniquement la couleur des sommets de la facette.

Pour calculer la couleur d'un sommet en utilisant le modèle d'illumination du Z-buffer, on a besoin de la normale à l'objet en ce sommet. On définit alors cette normale

en fonction des normales des facettes qui touchent ce sommet (par exemple en faisant la moyenne des normales de ces facettes).

Une fois calculée la couleur des sommets de la facette, on fait une interpolation linéaire des couleurs des sommets pour en déduire la couleur de chaque point de la facette.

Lissage de Phong

Le lissage de Phong fonctionne sur le même principe que le lissage de Gouraud, mais au lieu d'interpoler la couleur des sommets, on interpole la valeur de la normale en chaque sommet pour en déduire la valeur de la normale en chaque point de la facette.

Une fois qu'on connait la normale en chaque point de la facette, on peut y appliquer le modèle d'illumination du Z-buffer.

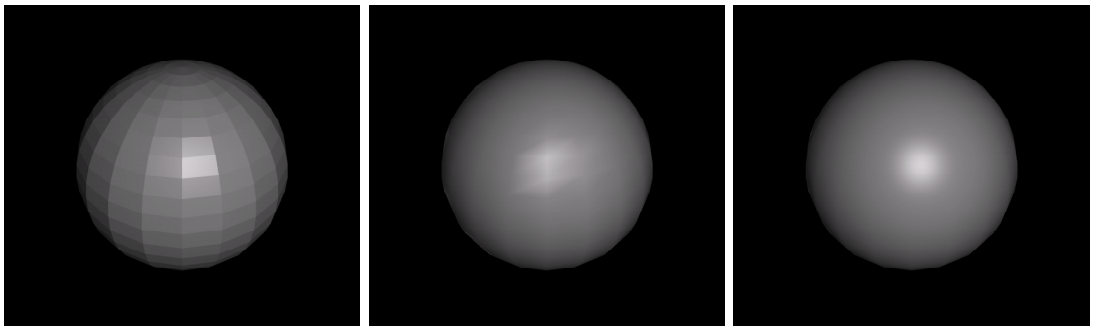


FIG. 1.5 – Objet non lissé, lissage de Gouraud et lissage de Phong

Comparaison lissage de Gouraud / lissage de Phong

Le lissage de Phong offre un plus joli rendu : les dégradés de couleurs sont plus fluides et se rapprochent plus de la réalité (le lissage de Gouraud a tendance, dans certains cas, à gommer des maxima lumineux).

Cependant, pour d'autres modèles d'illumination, le lissage de Gouraud peut s'averer meilleur. En particulier, dans les méthodes de radiosité, on ne pourra utiliser que le lissage de Gouraud.

1.1.5 Les faiblesses du Z-buffer

Le Z-buffer est un algorithme très efficace qui permet l'affichage rapide d'une scène. C'est pourquoi il est très souvent repris par d'autres méthodes de rendu qui l'utilisent pour l'affichage mais qui n'utilisent pas son modèle d'illumination.

En effet le modèle d'illumination du Z-buffer est très pauvre : il ne permet pas de représenter :

- les ombres
- les objets réflechissants (mirroirs...)

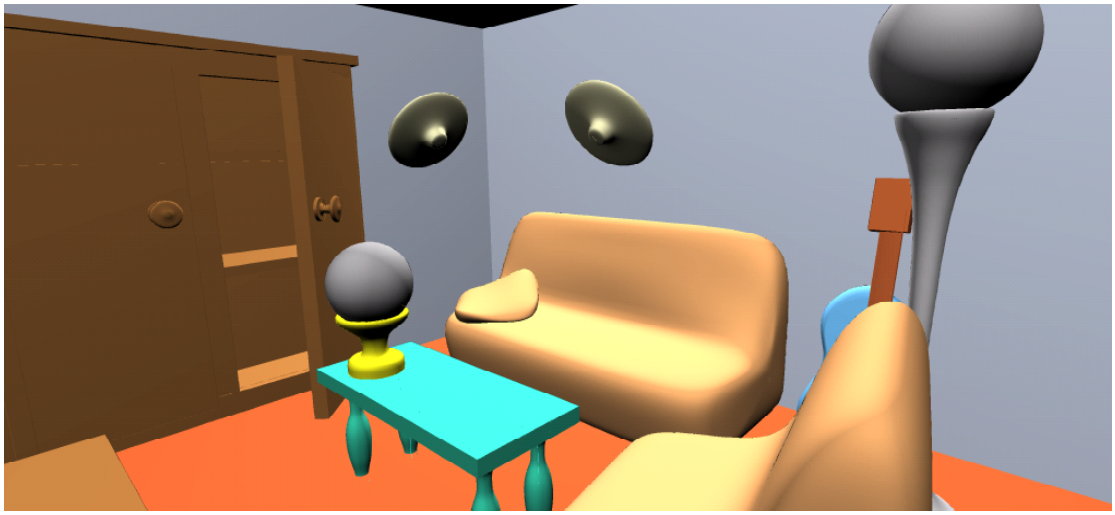


FIG. 1.6 – Un exemple de scène obtenue par Z-buffer

– les objets transparents

Chapitre 2

La radiosité discrète

La méthode de radiosité discrète est une variante de la méthode de radiosité classique (étudiée en annexe B), mise au point par le LLAIC1 sous la direction de Rémy Malgouyres. Dans cette méthode, on discrétise l'équation de base de la radiosité afin de remplacer le calcul d'intégrales par des opérations arithmétiques sur les entiers. On espère ainsi rendre la technique plus performante.

2.1 L'équation de base

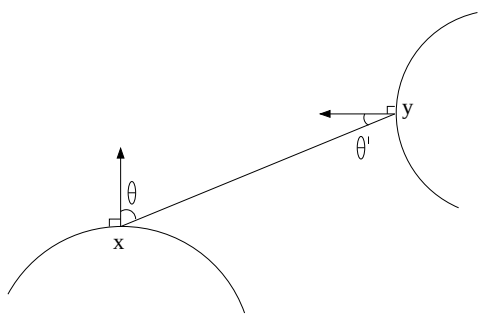


FIG. 2.1 – Les angles θ et θ'

La radiosité est basée sur une équation de conservation de l'énergie. Cette équation exprime le fait que l'intensité lumineuse émise par un point de la scène est proportionnelle à la somme de toutes les intensités reçues par ce point. On fait de plus l'hypothèse d'un rayonnement diffus idéal : chaque point de la scène emmet sa lumière dans toutes les directions avec la même intensité.
On note $B(x)$ la luminosité du point x de la scène. L'équation du rayonnement diffus est alors :

$$B(x) = E(x) + \rho(x) \int_{y \in scene} B(y) \frac{\cos\theta \cos\theta'}{\pi r^2} V(x,y) dy$$

où :

- $E(x)$ est *l'émmitance propre* de x. Ce terme est non nul si x est une source lumineuse.
- $V(x, y)$ vaut 1 si y est visible depuis x, c'est à dire si le segment $[x, y]$ n'est coupé par aucun objet de la scène. $V(x, y)$ vaut 0 sinon.
- $\rho(x)$ est le coefficient de réflection en x
- θ et θ' sont définis sur la fig.2.1.

2.2 Discrétisation de la scène 3D

La première étape de cette méthode est de discrétiser la scène, c'est à dire de transformer les polyèdres qui composent la scène en un ensemble de *voxels*.

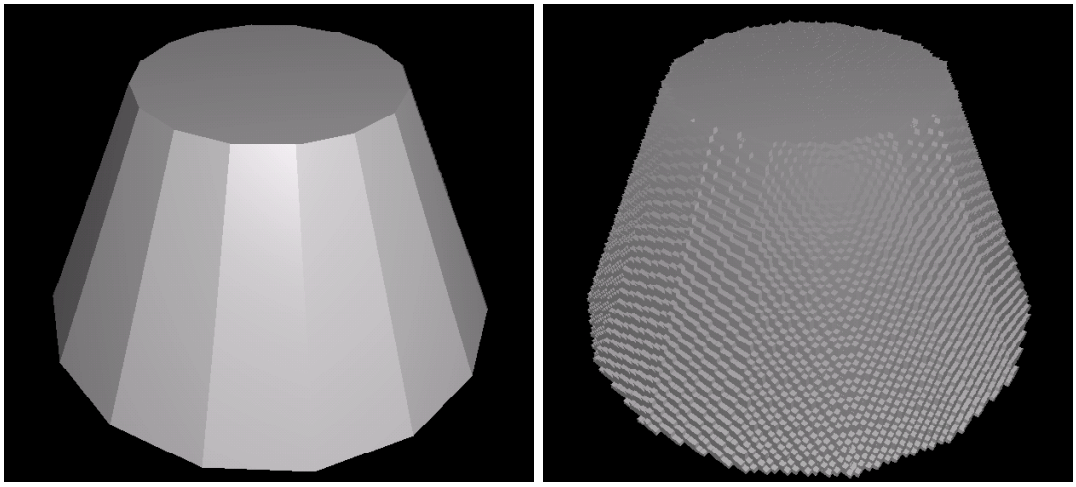


FIG. 2.2 – discrétisation d'un polyèdre

2.2.1 Les structures de données

Les voxels

Les voxels sont les homologues 3D des pixels. Ce sont des points à coordonnées entières $(i, j, k) \in \mathbb{Z}^3$. Un voxel $v = (i, j, k)$ est représenté par un petit cube de coté 1 centré sur le point (i, j, k).

Octree

Une scène peut se représenter par une matrice 3D de booléens. Cependant, cette solution n'est guère économique en termes d'espace. C'est pourquoi on choisit une structure arborescente : l'*octree*.
Un octree est un arbre dont chaque nœud a au plus huit fils. La racine de l'octree représente la scène toute entière. Un nœud représente un cube dans l'espace, et les fils

de ce nœud représente chacun un huitième de ce cube. Cette structure a l'avantage de compresser l'information.

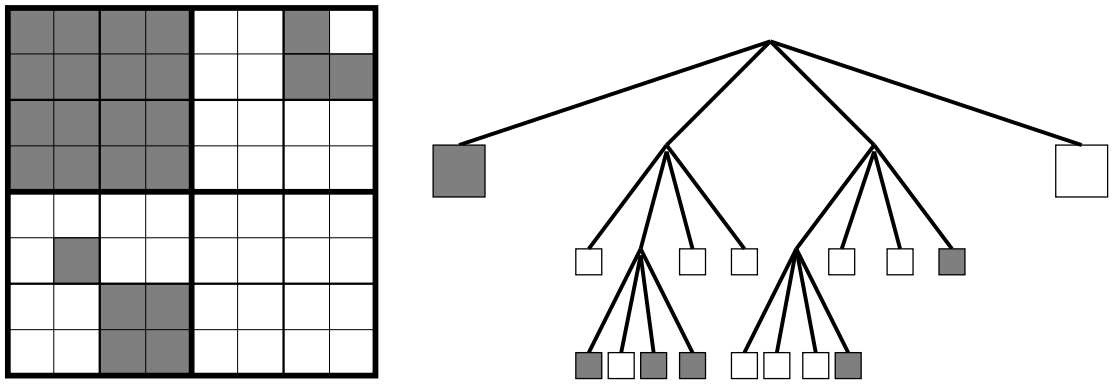


FIG. 2.3 – Une image en 2 dimensions et le quadtree correspondant

La figure 2.3 illustre cette construction en dimension 2. Dans cette dimension, cette structure s'appelle un quadtree mais cela fonctionne de la même manière. Le premier fils représente le carré en haut à gauche de la zone, le deuxième représente le carré en haut à droite, le troisième représente le carré en bas à gauche et le quatrième fils représente le carré en bas à droite.

2.2.2 La n-connexité

On dit que deux voxels v et v' sont :
- 6-voisins s'ils ont une face en commun et que $v \neq v'$;
- 18-voisins s'ils ont une arrête en commun et que $v \neq v'$;
- 26-voisins s'ils ont un sommet en commun et que $v \neq v'$.

Pour $n \in \{6, 18, 26\}$, on dit que v et v' sont n-connectés dans l'objet X s'il existe une suite finie $(v_0 = v, v_1, \ldots, v_p = v')$ de voxels de X tels que pour tout $i \in \{0, \ldots, p-1\}$, v_i et v_{i+1} sont n-voisins.

La relation *être n-connectés dans l'objet X* est une relation d'équivalence. Ses classes d'équivalences sont appellées les composantes n-connexes de l'objet.

2.2.3 Présentation du problème

On dispose d'une scène composée de polyèdres à sommets dans $\mathbb{R}^3$ et on souhaite approximer chacun de ces polyèdres par un ensemble de voxels qui 6-déconnecte l'interieur et l'exterieur du polyèdre.

2.2.4 Algorithme de discrétisation

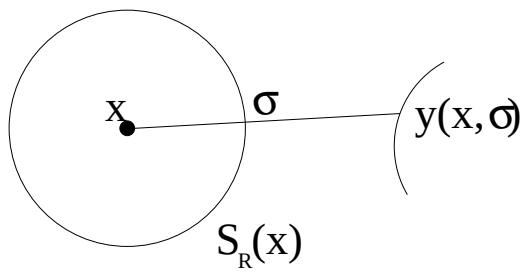
Pour chaque facette f de la scène :
On projette f sur un des plans Oxy, Oxz ou Oyz
On applique un algorithme de remplissage sur la projection $p(f)$ de f pour déterminer quels pixels sont à l'interieur $p(f)$
Pour chaque pixel x de l'interieur de $p(f)$,
On calcule la troisième coordonné de x grâce à l'équation du plan de la facette f. On obtient ainsi un voxel v.
On insère v dans l'octree

2.3 Résolution de l'équation

2.3.1 Discrétisation de l'équation

Un calcul préliminaire

On note :
- $S_R(x)$ la sphère de centre x et de rayon $R \in \mathbb{R}$;
- pour $\sigma \in S_R(x)$, $y(x,\sigma)$ le premier point rencontré par la demi-droite d'origine x et passant par σ.



On a alors, par changement de variable :

$$\begin{aligned}\int_{y\in scene} B(y)\frac{\cos\theta\cos\theta'}{\pi r^2}V(x,y)dy &= \int_{\sigma\in S_R(x)} B(y(x,\sigma))\cos\theta\left(\cos\theta'\frac{d(y(x,\sigma))}{\pi\|x-y(x,\sigma)\|^2}\right)\\ &= \int_{\sigma\in S_R(x)} B(y(x,\sigma))\frac{\cos\theta}{\pi R^2}d\sigma\end{aligned}$$

Calcul de $y(x,\sigma)$

Le calcul de l'intersection d'un rayon et d'un objet de forme quelconque est très coûteux. Par contre il est relativement plus simple de savoir si un rayon rencontre une sphère ou un parallélépipède rectangle, c'est pourquoi on associe à chaque objet de la scène une boîte englobante (*bounding box*).

Cette boîte englobante contient l'objet et possède une forme simple telle qu'il est facile de savoir si un rayon l'intersecte. Ainsi on commence par regarder si le rayon

rencontre la boîte englobante, puis on ne réalise le calcul d'intersection rayon-objet que si le rayon coupe la boîte englobante.

On se sert de la structure d'octree pour accélérer la recherche : on parcourt le rayon en partant de x en testant à chaque pas s'il y a intersection avec un objet de la scène. Si on voit que le rayon entre dans une zone de l'octree qui ne contient pas de voxel, on ne fait pas le calcul d'intersection dans cette zone et mais on continue le calcul directement à la sortie de la zone.

La méthode de la sphère discrète

On définit la sphère discrète $\Sigma_R(x)$ par :

$$\Sigma_R(x) = \{\text{voxel } v / v \in B_R(x) \text{ et } v \text{ a un 6-voisin hors de } B_R(x)\}$$

où $B_R(x)$ est la boule de centre x et de rayon R.

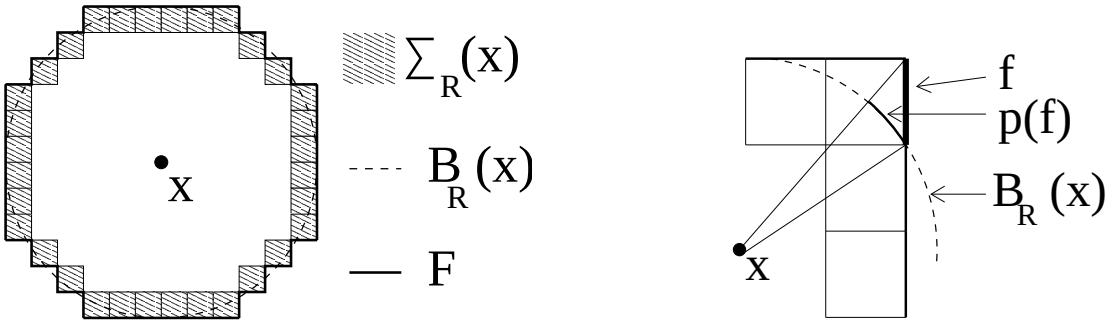


FIG. 2.4 – Méthode de la sphère discrète

On appelle *surfel* une face d'un voxel et on définit l'ensemble F des surfels qui définissent la frontière exterieure de $\Sigma_R(x)$:

$$F = \left\{ f \text{ surfel entre } v \text{ et } v' / v \in \Sigma_R(x) \text{ et } v' \notin B_R(x) \right\}$$

Enfin , pour $f \in F$, on note $p(f)$ la projection de f sur $S_R(x)$ (voir fig.2.4). En approximant l'intégrale au moyen d'une somme de Riemann, on obtient :

$$\begin{aligned} \int_{\sigma \in S_R(x)} B(y(x,\sigma)) \frac{\cos\theta}{\pi R^2} d\sigma &= \sum_{f \in F} \int_{\sigma \in p(f)} B(y(x,\sigma)) \frac{\cos\theta}{\pi R^2} d\sigma \\ &\simeq \sum_{f \in F} A(f) B(y(x, v_f)) \frac{\cos\theta}{\pi R^2} \end{aligned}$$

où :

- v_f est le voxel de $\Sigma_R(x)$ dont f est une face
- $A(f)$ est l'angle solide sous lequel est vu f depuis x (c'est l'aire de $p(f)$ divisée par πR^2).

2.3.2 Méthodes de Jacobi et de Gauss-Siedel

Si on note $\varphi(B)(x) = E(x)+\rho(x)\int_{\sigma\in S_R(x)} B(y(x,\sigma))\frac{\cos\theta}{\pi R^2}d\sigma$, on constate que résoudre l'équation de la radiosité revient à trouver un point fixe de φ.
Or, φ est contractante (au sens de la norme infinie).

Preuve :

$$\begin{aligned}
\|\varphi(B)(x) - \varphi(B')(x)\| &= \left\|\rho(x)\int_{\sigma\in S_R(x)} \left(B(y(x,\sigma)) - B'(y(x,\sigma))\right)\frac{\cos\theta}{\pi R^2}d\sigma\right\| \\
&\leq \|\rho\|_\infty \left|\int_{\sigma\in S_R(x)} \frac{\cos\theta}{\pi R^2}d\sigma\right| \|B - B'\|_\infty \\
&= \|\rho\|_\infty \|B - B'\|_\infty
\end{aligned}$$

On a donc $\|\varphi(B) - \varphi(B')\|_\infty < \|B - B'\|_\infty$, donc φ est contractante.□

Il y a donc une unique solution à l'équation $\varphi(B) = B$.
Désormais, on va faire l'approximation $\varphi(B) \simeq \psi(B) = E(x)+\sum_{f\in F} A(f)B(y(x,v_f))\frac{\cos\theta}{\pi R^2}$.
Si on note $(x_i)_{i=1..N}$ les voxels de la scène, l'équation de la radiosité discrète peut se noter de façon simplifiée :

$$B(x_i) = \psi(B)(x_i) = E(x_i) + \sum_{j=1}^{N} a_{i,j}B(x_j)$$

Méthode de Jacobi

On cherche une fonction B point fixe de ψ. B associe à chaque voxel de la scène (ensemble fini de voxels) une couleur.
ψ est contractante et l'ensemble dans lequel on recherche B est complet (car on peut identifier B à un vecteur en dimension finie), donc la suite définie par :

- B_0 est constante nulle
- $B_{k+1}(x_i) = \psi(B_k)(x_i) = E(x_i) + \sum_{j=1}^{N} a_{i,j}B_k(x_j)$

converge vers B.

méthode de Gauss-Siedel

Si on cherche à implémenter la méthode de Jacobi, on a besoin de deux tableaux pour faire le calcul $B_{k+1}(x_i) = E(x_i) + \sum_{j=1}^{N} a_{i,j}B_k(x_j)$: un pour stocker B_k et l'autre pour stocker B_{k+1}.
Il est possible d'économiser de l'espace en ne gardant en mémoire qu'un seul tableau : on utilise le même tableau pour stocker B_k et B_{k+1}. Les valeurs de B_k sont alors écrasées au fur et à mesure par celles de B_{k+1}. On a alors :

$$B_{k+1}(x_i) = E(x_i) + \sum_{j=1}^{i-1} a_{i,j}B_{k+1}(x_j) + \sum_{j=i+1}^{N} a_{i,j}B_k(x_j)$$

FIG. 2.5 – Affichage de la radiosité par le Z-buffer

Cette méthode est plus simple à implémenter, requiert moins d'espace mémoire et il est démontré que la suite (B_k) ainsi définie converge également vers B.

2.4 Affichage

Pour l'affichage de la scène, soit à l'écran, soit dans un fichier de type bitmap (BMP), on a besoin de connaître la couleur des sommets des facettes. Le sommet d'une facette est un point de $\mathbb{R}^3$, il ne coincide donc en général pas avec un voxel. Pour déterminer sa couleur, on réalise donc une moyenne des couleurs des voxels qui l'entourent. Ensuite, on fait un lissage de Gouraud (i.e. on interpole la couleur d'un point en fonction de la couleur des sommets de la facette sur laquelle il se trouve : voir 1.1.4) sur les données calculées par la radiosité discrète, puis on utilise une variante de l'algorithme du Z-buffer ou bien une variante du lancer de rayons :

Une variante du Z-buffer

On fait comme pour l'algorithme d'affichage du Z-buffer, mais, au lieu d'utiliser son modèle d'illumination pour calculer la couleur des points, on utilise les données calculées par la méthode de radiosité discrète. Cette méthode à l'avantage d'être très rapide (elle permet une navigation interactive dans la scène).

Une variante du lancer de rayons

La méthode du lancer de rayons est étudiée en annexe A. On lance un rayon par pixel comme dans la technique classique de lancer de rayons. Lorsqu'un rayon rencontre un objet, on calcule un rayon de transparence et un rayon de reflection, mais on ne calcule pas les rayons d'éclairement : ils sont remplacés par les données calculées par la méthode de radiosité discrète. Le calcul d'une image par ce procédé est très lent car les temps de calcul de la radiosité et du lancer de rayons s'accumulent. Cependant le rendu est très fidèle à la réalité car les deux méthodes se complètent.

Le seul inconvénient qui persiste dans cette technique est que la transparence et l'aspect réflechissant des objets n'est pas pris en compte dans le calcul de l'éclairement des objets qui les entourent.

FIG. 2.6 – Affichage de la radiosité par lancer de rayons

2.5 Comparaison avec le lancer de rayon

Avantages par rapport au lancer de rayon :
- Meilleur rendu des réflections diffuses entre objets
- Le calcul des intensités lumineuses ne dépend pas du point de vue : il est possible de naviguer dans la scène en temps réel.

Inconvénient par rapport au lancer de rayon :
La méthode de base ne permet pas de représenter d'objets transparents ou brillants. Cependant, la combinaison de la radiosité et du lancer de rayon permet de représenter de tels objets. On perd alors l'indépendance du calcul par rapport au point de vue.

L'inconvénient majeur (commun à la radiosité et au lancer de rayons) est que les propriétés réfléchissantes ou transparentes des objets ne sont pas prises en compte pour calculer l'éclairement des autres objets de la scène. Ceci entraîne en particulier que l'ombre des objets transparents n'est pas réaliste.
Pour corriger ce défaut on peut mettre en oeuvre des techniques de lancer de rayons *en partant des sources* plutôt qu'en partant de l'observateur (technique de *Monte Carlo* par exemple). S'il est possible de combiner ces méthodes avec la radiosité, elles sont cependant extrèmement couteuses.

Chapitre 3

Subdivision adaptative des facettes

Ce chapitre s'inscrit dans le cadre de la radiosité discrète.
Tous les objets 3D de la scène à représenter sont des polyèdres.

Pour un aspect plus lisse de la scène, on ne calcule pas la couleur de chaque point du polyèdre : on se contente de calculer la couleur de chacun des sommets puis on réalise un lissage de Gouraud sur chaque facette (voir paragraphe 1.1.4).

La taille des facettes joue alors un rôle très important :

- Si on utilise de trop grandes facettes, on ne voit plus les details : ils sont atténués par le lissage de Gouraud (voir fig 3.1)
- Si on utilise de trop petites facettes dans des zones ou la couleur est presque uniforme, le lissage ne sera pas satisfaisant. De plus, le nombre des facettes fera que le temps d'affichage deviendra beaucoup trop lent pour une navigation agréable dans la scène, et si on combine le lancer de rayon avec la radiosité discrète, un trop grand nombre de facettes ralentit énormément le calcul du rendu.

Le problème est donc de calculer une taille optimale pour chaque facette de la scène.

Il est évident que la taille opimale ne sera pas la même pour toutes les facettes de la scène : les zones comportant de petits détails nécessitent de petites facettes, tandis que les zones uniformément colorées ne requièrent que peu de facettes.

FIG. 3.1 – Image non lissée, grandes fracettes et image trop lissée

On opte donc pour un algorithme de subdivision adaptative des facettes. Cet algorithme fonctionne en deux étapes :

- On transforme les polyèdres pour n'avoir plus que des facettes triangulaires.
- On réalise la subdivision adaptative de ces facettes triangulaires.

Mon travail consistait à mettre au point puis à implémenter cet algorithme.

3.1 Triangulation des facettes

Le but de cette étape de l'algorithme est de découper des facettes de forme quelconque en facettes triangulaires.

La principale difficulté provient du fait que les facettes qu'on doit découper ne sont pas nécessairement convexes. De plus, on souhaite obtenir un algorithme le plus rapide possible.

3.1.1 Généralités sur les facettes

Une facette est un polygone contenu dans un plan. On la représente par la liste $(P_1, P_2, ..., P_n)$ de de ses sommets. Les arêtes de la facette sont les segments $[P_i, P_{i+1}]$ pour $i = 1..n-1$ ainsi que le segment $[P_n, P_1]$.

On dit qu'un point A est à *l'intérieur* de la facette si une demi-droite issue de A coupe un nombre impair d'arêtes de la facette (voir fig 3.2). Cette définition est bien cohérente car on peut montrer que la parité du nombre d'arêtes coupées ne dépend pas de la demi-droite issue de A considérée.

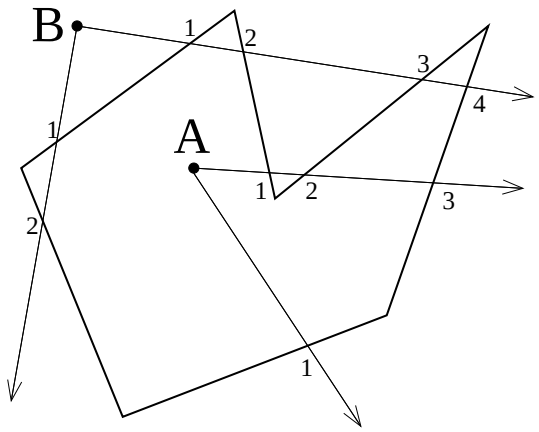


FIG. 3.2 – A est à l'interieur, mais pas B

3.1.2 Algorithme de triangulation

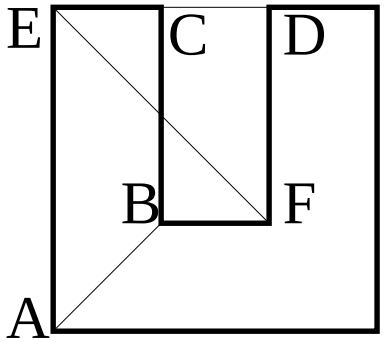


FIG. 3.3 – AB convient, mais pas CD ni EF

Voici les principales étapes de l'algorithme :
Pour chaque facette f :
Si f possède exactement 3 sommets, on garde f inchangée. Sinon :

On projette f sur un des plans Oxy, Oxz ou Oyz. On obtient ainsi un polygone P.
On cherche un couple (A, B) de sommets de P qui vérifie :
– le segment $[A, B]$ ne coupe aucune des arêtes de P.
– le segment $[A, B]$ est à l'interieur de P.
Cette étape est illustrée sur la figure 3.3.
On rajoute l'arête $[A, B]$ à f, ce qui coupe f en deux facettes f_1 et f_2. On applique ensuite l'algorithme récursivement à f_1 et f_2.

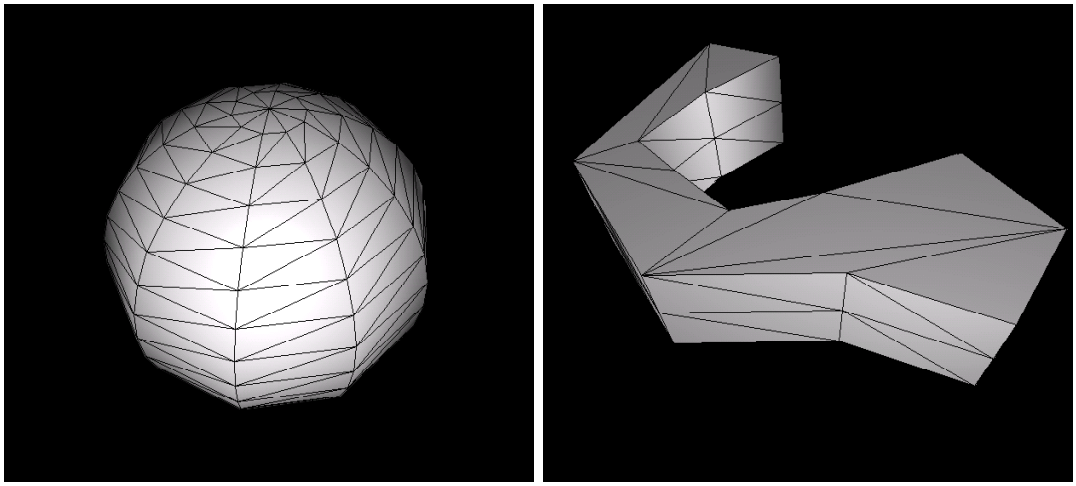


FIG. 3.4 – Triangulation de facettes

3.1.3 Détail de l'algorithme

On va maintenant décrire plus précisément comment réaliser de manière rapide les principales étapes de cet algorithme.

Détermination du plan de projection

On souhaite projeter une facette sur un des plans Oxy, Oxz ou Oyz. On veut de plus que l'angle entre la facette et le plan de projection ne soit pas trop près de l'angle droit afin que la facette ne soit pas "écrasée".

On souhaite de plus ne réaliser que des opérations simples sur les coordonnées des sommets de la facette (pas de calcul de racines carrées ...).

Pour ce faire, on cherche un couple $(\overrightarrow{a}, \overrightarrow{b})$ d'arêtes de la facette tel que les vecteurs $\overrightarrow{a}$ et $\overrightarrow{b}$ ne soient pas colinéaires. Pour éviter les problèmes d'arrondi, on demande même que l'angle formé par ces vecteurs soit supérieur à un angle θ_{min}. Ceci équivaut à la condition :

$$\|\overrightarrow{a} \wedge \overrightarrow{b}\| \geq \|\overrightarrow{a}\|\|\overrightarrow{b}\| \sin\theta_{min}$$

En pratique, on choisit $\sin^2\theta_{min} = \frac{1}{10}$. En élevant la condition précédente au carré, on obtient une nouvelle condition qui ne met en jeu que des opérations simples sur les coordonnées des sommets de la facette :

$$10\|\overrightarrow{a} \wedge \overrightarrow{b}\|^2 \geq \|\overrightarrow{a}\|^2\|\overrightarrow{b}\|^2$$

On parcourt tous les couples $(\overrightarrow{a}, \overrightarrow{b})$ jusqu'à ce que cette condition soit vérifiée.
Si aucun couple $(\overrightarrow{a}, \overrightarrow{b})$ ne vérifie l'inégalité demandée, on prend celui pour lequel l'angle formé par les vecteurs $\overrightarrow{a}$ et $\overrightarrow{b}$ est le plus grand.
Une fois $\overrightarrow{a}$ et $\overrightarrow{b}$ déterminés, il n'y a plus qu'à projeter sur le plan orthogonal au vecteur qui porte la plus grande composante de $\overrightarrow{a} \wedge \overrightarrow{b}$.

Déterminer si un segment est à l'interieur de la facette

On sait que le segment ne coupe aucune arête de la facette. Il est cependant possible qu'il soit à l'exterieur de la facette (voir segment CD, fig.3.3).

Une première possibilité :
On peut choisir un point du segment (par exemple son milieu) et regarder s'il est à l'interieur de la facette en comptant le nombre d'intersections entre une demi-droite issue de ce point et les arêtes de la facettes (cf. définition de l'interieur d'une facette au 3.1.1).

Cependant, cette méthode s'avère couteuse parce que cette procédure est appelé un grand nombre de fois pour chaque facette à subdiviser (car l'algorithme de triangulation est récursif).

Une méthode plus efficace :
On commence par calculer *l'orientation* du polygone P. En effet un polygone est orienté

car il est défini par la suite de ses sommets $(P_0, ..., P_n)$: le sens dans lequel sont parcouru ses sommets peut être le sens trigonométrique ou le sens anti-trigonométrique. Une fois l'orientation du polygone déterminée, étant donnée une arête de P, on sait si l'interieur de P est à droite ou à gauche de cette arête. Ainsi, pour savoir si un segment $[P_i, P_k]$ est à l'interieur de P, il suffit de vérifier que l'angle $\widehat{P_{i-1}P_iP_k}$ est bien compris entre 0 et $\widehat{P_{i-1}P_iP_{i+1}}$, ce qui se fait de façon peu couteuse en utilisant des produits scalaires (voir fig 3.5).

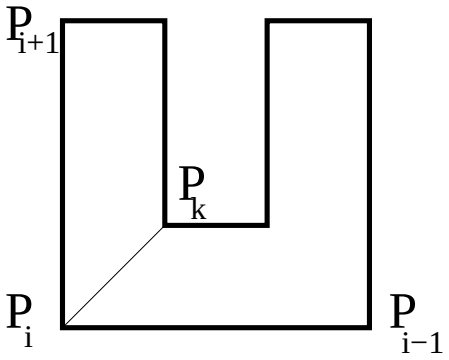


FIG. 3.5 – Division d'une facette

Pour calculer l'orientation de P, on se sert du fait que la somme des angles d'un polygone est du type $k\pi$, avec $k \in \mathbb{Z}$ (pour le voir il suffit de décomposer cette somme en une somme d'angles de triangles, cf. fig.3.6). Le signe de k détermine alors l'orientation du polygone. Ce calcul est couteux du fait de l'emploi de la fonction `acos`, cependant on ne le fait qu"une seule fois pour chaque facette de la scène car on s'arrange pour conserver l'orientation des polygones lors des appels récursifs de l'algorithme. Au total, cette méthode est donc plus efficace que la précédente.

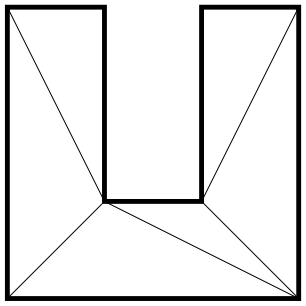


FIG. 3.6 – La somme des angles d'un polygone est la somme des angles de triangles

3.2 Subdivision des facettes

Tous les objets de la scène ont maintenant des facettes triangulaires. Il faut subdiviser ces facettes dans les zones qui présentent de fortes variations de couleur.

Pour savoir si un facette doit être divisée, on calcule la couleur de la facette en des *points de contrôle*. Ces points de controle sont répartis uniformément sur toute la surface de la facette. A partir des données ainsi obtenues on regarde si la facette vérifie un *critère de division*, qui peut dépendre de :

- l'aire de la facette
- la différence de couleur maximum entre deux points de contrôle
- la distance entre les deux points de contrôle qui ont les couleurs les plus differentes

Ensuite, il faut determiner de quelle façon doit être divisée la facette. Cela peut dépendre de la forme de la facette et de la position des deux points de contrôle qui ont les couleurs les plus differentes.
Une fois que la facette a été divisée, on applique récursivement l'algorithme aux facettes obtenues.

3.2.1 Contraintes

La forme des facettes

Une facette qui a une forme très allongée (triangle presque plat) peut avoir une très grande longueur mais une toute petite surface qui fait qu'elle ne sera pas subdivisée. Les facettes de ce type doivent pourtant être évitées car elles dégradent la qualité de l'image : l'objet apparaît strié.

Pour éviter ce problème on peut :

- prendre en compte la forme de la facette dans le critère de division, de façon à favoriser la division des facettes allongées
- ou bien eviter de creer des facettes allongées

C'est la deuxième solution qui a été retenue : on décide de toujours diviser une facette de manière à couper en deux par son milieu le côté le plus long du triangle. Cette façon de procéder n'est pas optimale si on souhaite avoir un nombre minimal de facettes à la fin de l'algorithme (on pourrait par exemple obliger les deux points de contrôle ayant les couleurs les plus différentes à ne pas se retrouver dans la même sous facette), mais elle élimine les facettes trop allongées.

Précision du calcul de l'ordinateur

Supposons qu'on ait deux facettes $F = ABC$ et $F' = BCD$ qui se touchent par le coté BC et qu'on souhaite diviser la facette F en deux facettes $F_1 = ABE$ et $F_2 = AEC$ où E est sur le segment $[BC]$ (voir fig.3.7).
La précision du calcul fait que le point E peut ne pas tomber exactement sur le segment $[BC]$, mais un peu à coté.

- si E tombe à l'interieur de F', les facettes F_1 et F' ainsi que les facettes F_2 et F' vont se chevaucher légèrement, ce qui n'aura pas de conséquence visible.

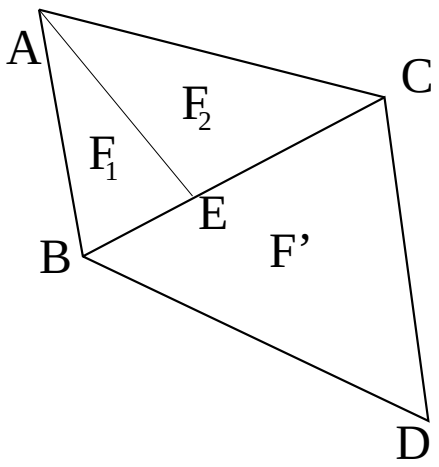


FIG. 3.7 – Division de la facette F

- si E tombe à l'interieur de F, c'est plus grave : il va y avoir un trou visible dans l'objet car les facettes ne seront pas en contact (voir fig.3.8).

Deux solutions ont été envisagées pour résoudre ce problème :

- s'arranger pour que E tombe toujours à l'interieur de F' et jamais à l'interieur de F en obligeant toutes les facettes de la scène à se chevaucher légèrement.
- diviser aussi F' en $F'_1 = BDE$ et $F'_2 = CDE$ pour que les facettes reposent sur les mêmes segments (voir fig.3.9).

La deuxième solution, plus propre mais plus complexe à mettre en œuvre, a été adoptée. Elle nécessite la construction, et la mise à jour au fur et à mesure des subdivisions, d'un graphe. Les sommets de ce graphe représentent les sommets des facettes de la scène. Si deux sommets de la scène appartiennent à la même facette f, les sommets correspondant du graphe sont reliés par une arête et cette arête est étiquetée par f. Ainsi, chaque arête du graphe est étiquetée par au plus deux facettes. Un exemple d'un tel graphe est illustré à la figure 3.10.

3.2.2 Critère de division

Etude qualitative

Les paramètres que doit donner l'utilisateur du programme sont :

taille minimale des facettes : Pour s'assurer de la terminaison de l'algorithme de subdivision, on fixe une taille minimum en-dessous de laquelle on choisit toujours de ne pas diviser la facette. L'utilisateur peut la choisir : une petite taille minimum augmentera la précision du rendu mais fera croître le nombre de facettes.

limite petite/grande facette : On utilise un critère de division différent selon que la surface de la facette est grande ou petite. Pour les facettes considérées comme grandes, la

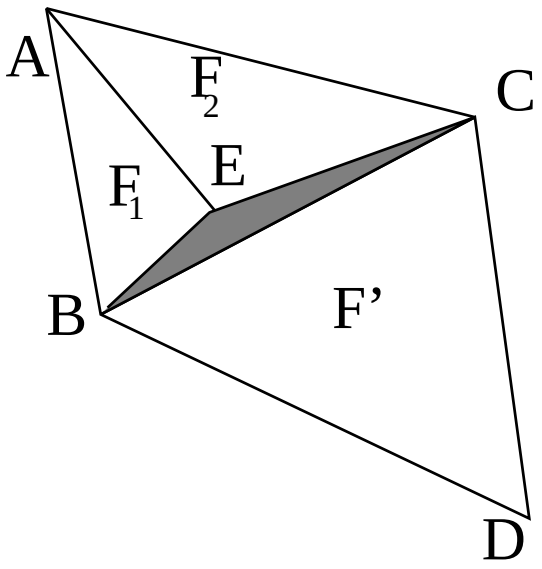


FIG. 3.8 – Des trous apparaissent dans l'objet

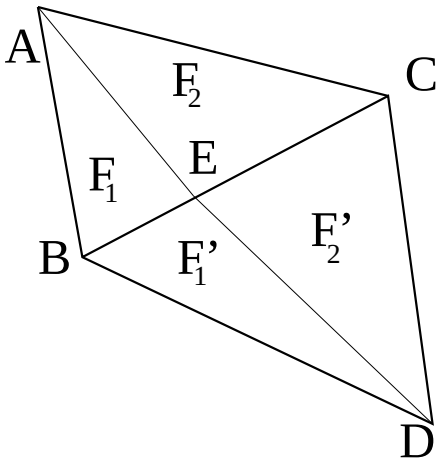


FIG. 3.9 – Division de la facette voisine

taille de la facette intervient dans le critère de division : plus la facette est grande et plus elle sera facilement divisée. En revanche, pour les petites facettes, on ne tient pas compte de la taille de la facette dans le critère de subdivision. Dans certains cas, il peut être souhaitable de ne pas faire de distinction entre ces deux types de facettes mais de traiter les facettes soit toutes comme des grandes, soit toutes comme des petites. Il suffit alors de régler la limite petite/grande facette à zéro ou à l'infini pour obtenir le résultat désiré.

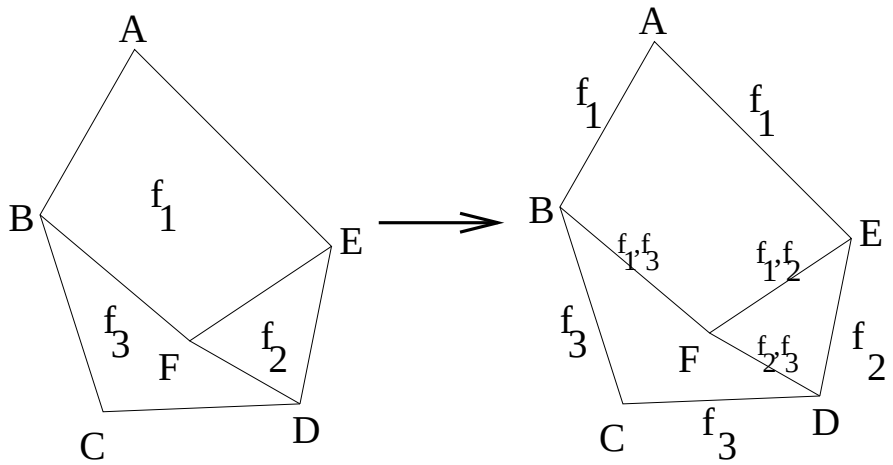


FIG. 3.10 – Un exemple de graphe des sommets

sensibilité : Plus la sensibilité de la méthode est importante, plus les facettes vont se diviser facilement. Accroître ce paramètre augmente donc la précision globale de l'image obtenue et aussi le nombre final de facettes.

densité des points de contrôle : Influe sur le nombre de points de contrôle dans chaque facette. Plus ce nombre est élevé, et moins un petit détail de l'image a de chances "d'échapper" à l'algorithme. Augmenter ce nombre rend la méthode plus précise mais plus coûteuse en temps.

Les figures 3.11 et 3.12 illustrent l'influence de ces paramètres. Ils sont notés sous la forme "X Y Z T" où X est la taille minimum des facettes, Y est la limite petite/grande facette, Z est la sensibilité et T est la densité des points de contrôle.

Choix d'une distance sur les couleurs

Une couleur X est codée par le triplet $(R_X, V_X, B_X) \in [0,1]^3$ de ses trois composantes : la rouge, la verte et la bleue. On cherche à définir une distance sur les couleurs.

La distance $d(X,Y) = max(|R_X - R_Y|, |V_X - V_Y|, |B_X - B_Y|)$ ne convient pas car il serait absurde que $d((R,V,B),(R+\epsilon, V-\epsilon, B)) = d((R,V,B),(R+\epsilon,V,B))$.

La distance $d(X,Y) = |R_X - R_Y| + |V_X - V_Y| + |B_X - B_Y|$ n'est pas non plus une bonne candidate. En effet, étant donné une couleur $X = (R_X, V_X, B_X)$, il semblerait normal que $d(X,(R_X+\epsilon, V_X+\epsilon, B_X+\epsilon)) < d(X,(R_X+3\epsilon, V_X, B_X))$ car la couleur $(R_X+\epsilon, V_X+\epsilon, B_X+\epsilon)$ est la couleur X en plus foncé tandis que $(R_X+3\epsilon, V_X, B_X)$ n'a rien a voir avec X. Or ceci n'est pas réalisé avec une telle distance.

C'est pourquoi on choisit la distance d_{coul} sur les couleurs, définie par :

$$d_{coul}(X,Y) = \sqrt{(R_X - R_Y)^2 + (V_X - V_Y)^2 + (B_X - B_Y)^2}$$

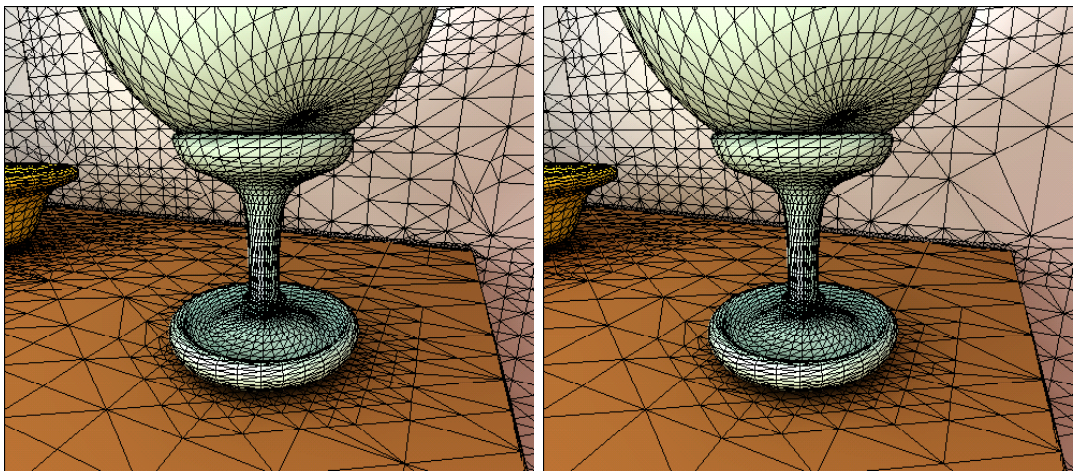


FIG. 3.11 – 1 0 50 3 et 1 0 100 3

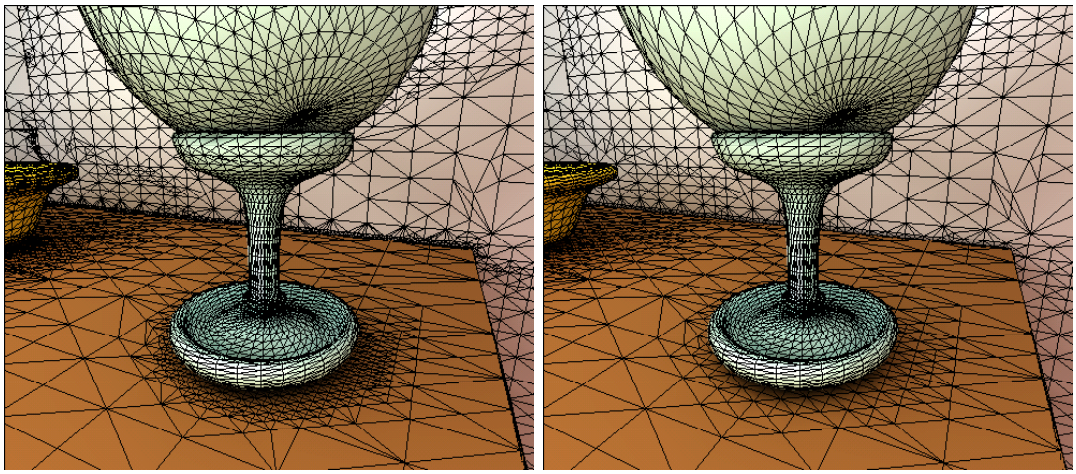


FIG. 3.12 – 1 30 50 3 et 15 0 50 3

Expression du critère

Etant donné une facette, on détermine les deux points de contrôle qui sont les plus éloignés au sens de d_{coul} et on note :

- d la distance entre les couleurs de ces deux points
- l la distance (géométrique) entre ces deux points
- l_0 le paramètre de limite petite/grande facette donné par l'utilisateur

Si la surface de la facette est inferieure à la taille minimale fixée par l'utilisateur, on ne divise pas.
Sinon, si $l > l_0$, la facette est grande. On la divise si $dl > s$ où s est la sensibilité de la méthode.

Si $l < l_0$, la facette est petite. On la divise si $dl_0 > s$.

3.2.3 Initialisation du graphe des sommets

On code le graphe sous forme d'un tableau de listes d'adjacence : chaque case du tableau correspond à un sommet du graphe et contient une liste de triplets (sommet adjacent, facette numero 1, facette numero 2).
Comme on va rajouter des sommets dans le graphe durant l'algorithme, il faut prévoir une structure de tableau à taille variable (on double la taille du tableau quand il est plein).
Pour ne pas stocker en double les informations, on oriente le graphe : les sommets étant numérotés, on ne représente que les arêtes qui partent d'un sommet vers un sommet qui a un numero superieur.

On ne dispose que d'un ensemble désordonné de facettes : on ne sait pas au début quelles sont les facettes qui se touchent. Il va donc falloir parcourir l'ensemble des facettes pour construire le graphe :
Pour chaque facette f :
Pour chaque couple de sommets (s, s') de f :
ajouter $\max(s, s')$ étiqueté par f dans la liste d'adjacence de $\min(s, s')$

3.3 Résultats expérimentaux

Cet algorithme de subdivision adaptative réduit considérablement le nombre de facettes de la scène, à qualité d'image équivalente.
Pour la scène de référence utilisée (fig.3.3), qui comporte 1 000 000 de facettes de même taille, on peut obtenir une image de qualité équivalente pour moins de 400 000 facettes, et pour 600 000 facettes on a un résultat beaucoup plus joli que la scène initiale.

Sur les figures 3.13,3.14 et 3.15, on voit que les améliorations se situent aussi bien dans les zones comportant de fortes variations de couleur (l'ombre de la lampe est plus régulière, la limite mur/table est plus précise) que dans les zones uniformes (le mur a un aspect plus lisse).

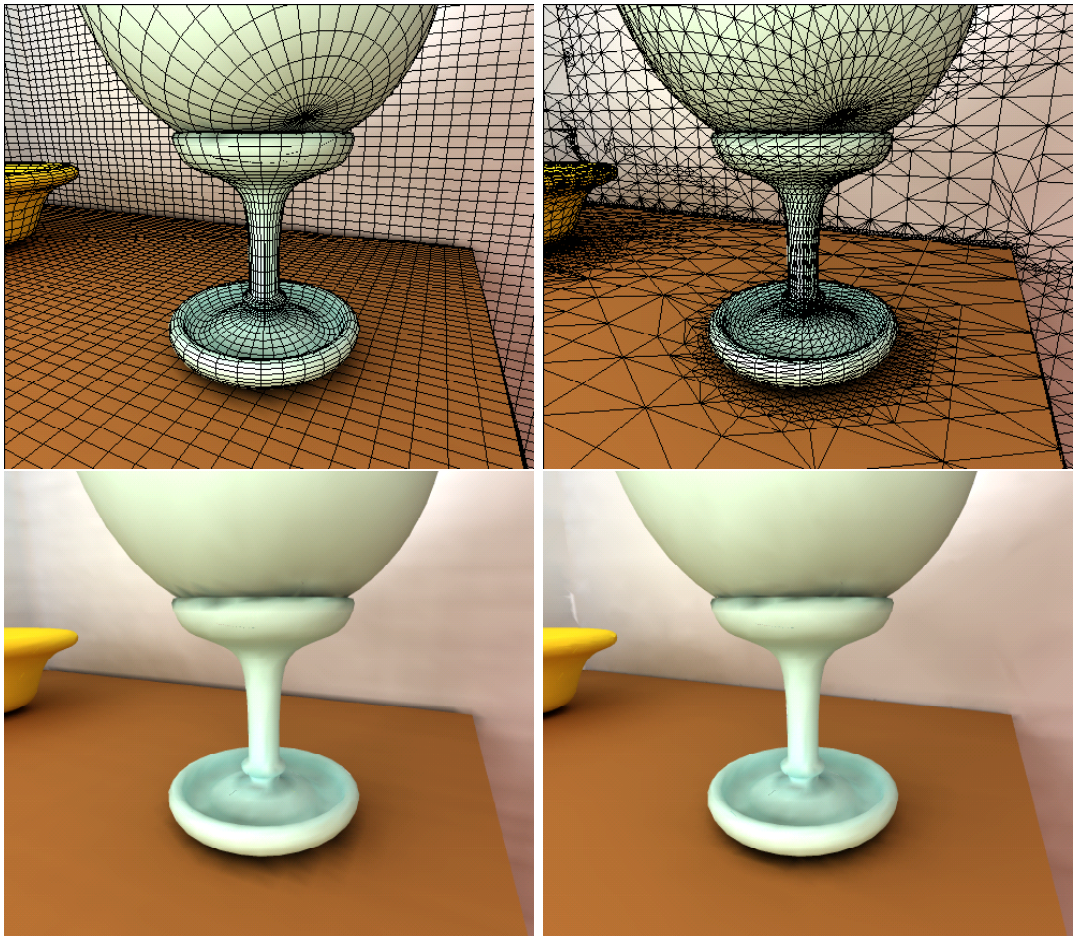


FIG. 3.13 – 1 000 000 de facettes rectangulaires / 600 000 facettes par subdivision adaptative

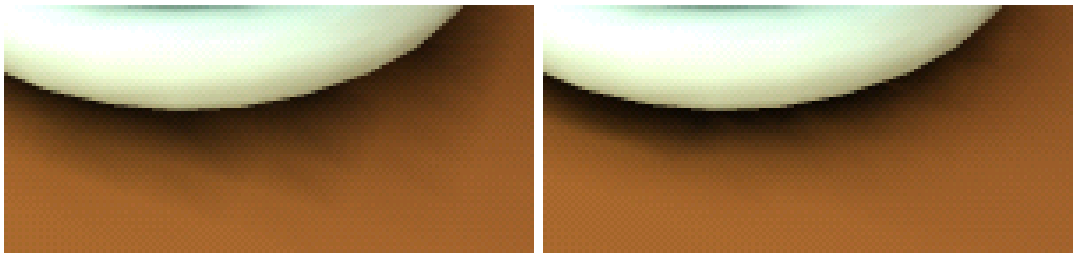


FIG. 3.14 – Détail de l'ombre

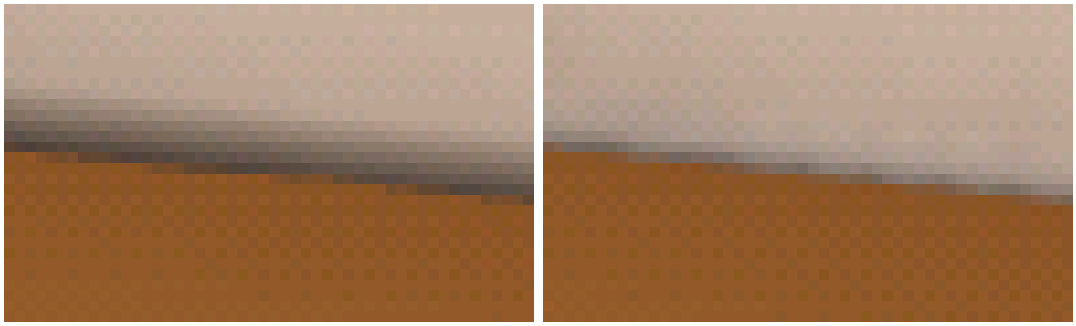


FIG. 3.15 – Limite mur / table

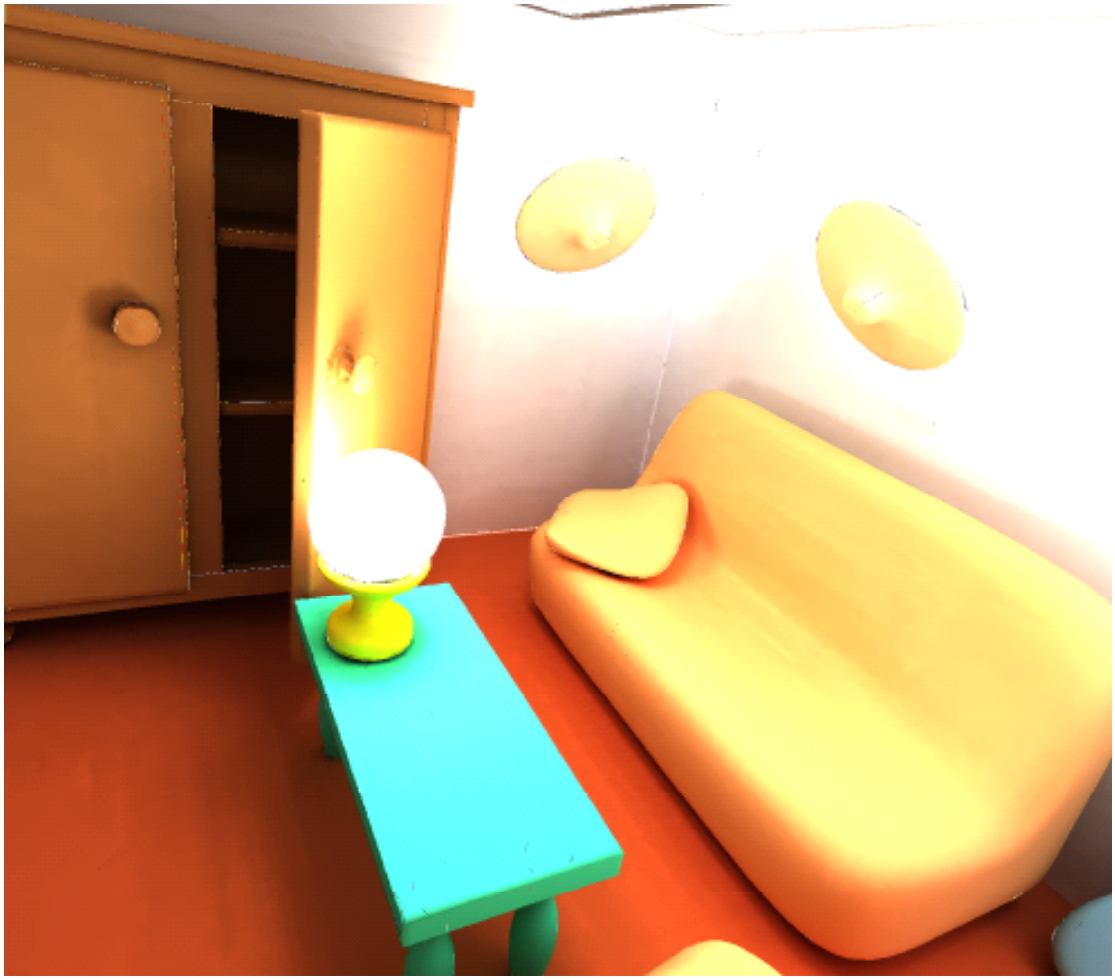


FIG. 3.16 – Une vue globale de la scène

Chapitre 4

Elimination des points simples non visibles

A l'issue des calculs de radiosité, on connaît la couleur des voxels, c'est à dire des points à coordonnées entières $(i, j, k) \in \mathbb{Z}^3$. Lorsque l'on souhaite afficher le résultat à l'écran, on a besoin de pouvoir calculer la couleur de n'importe quel point de $\mathbb{R}$ (voir paragraphe 2.4). La couleur d'un point à coordonnées non entières est alors définie comme la moyenne des couleurs des voxels voisins à ce point, et on espère que ces voxels sont sur la surface de l'objet.

Cependant, il arrive que des voxels voisins ne soient pas sur la surface de l'objet. Leur couleur n'a alors rien à voir avec la couleur qu'on souhaite calculer et ils faussent l'image.

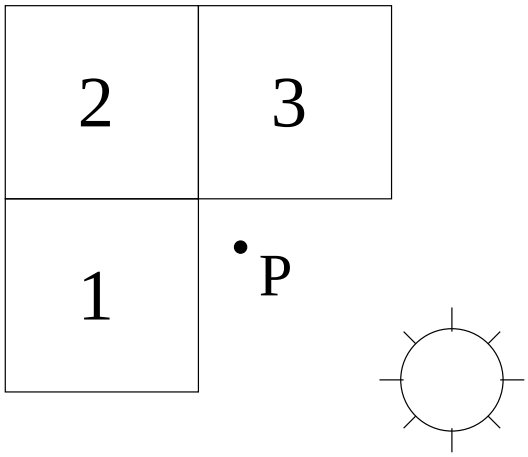


FIG. 4.1 – Le voxel 2 est simple

Un exemple de tel voxel est donné sur la figure 4.1 : on souhaite calculer la couleur du point P, on fait donc la moyenne des couleurs des voxels 1,2 et 3. Mais le voxel 2 est

noir car les voxels 1 et 3 le mettent à l'ombre de la source lumineuse, par conséquent la couleur du point P va apparaître sur l'image plus sombre qu'elle ne le devrait. Cela provoque en particulier la présence de lignes noires dans les angles de la pièce (voir fig 4.2).

FIG. 4.2 – Des traits noirs apparaissent dans les angles des murs et du plafond

Pour corriger ce problème on cherche à supprimer les voxels qui ne font pas réellement partie de la surface des objets. Ces voxels sont dits *simples* (ou points simples). Cependant, certains points simples ne créent pas de ligne noire : seuls les points simples non visibles (à "l'intérieur" des objets) posent problème. On souhaite donc conserver les points simples visibles, afin de modifier le moins possible la forme des objets.

4.1 Les points simples

Intuitivement, les points simples sont les voxels que l'on peut supprimer sans changer la topologie de l'objet, c'est à dire :

- sans déconnecter l'objet
- sans creer ou déplacer un trou dans la surface de l'objet

Plus formellement, pour $(n, \overline{n}) \in \{(6, 26), (26, 6)\}$, un voxel x de l'objet X est n-simple si :

- X et $X \setminus \{x\}$ ont le même nombre de composantes n-connexes
- $\overline{X}$ et $\overline{X} \cup \{x\}$ ont le même nombre de composantes $\overline{n}$-connexes
- Aucun trou de X n'est supprimé ni créé par effacement de x
- Aucun trou de $\overline{X}$ n'est supprimé ni créé par effacement de x

4.2 Détection des points simples

Les conditions de la définition des points simples n'étant pas facile à vérifier, on utilise une caractérisation plus efficace mais qui nécessite l'utilisation des *voisinages géodésiques*.

4.2.1 Voisinages géodésiques et nombre topologique

On commence par définir récursivement les ensembles $N_n^k(x, X)$ pour $k = 1..3$ et x un voxel de X par :
- $N_n^1(x, X) = \{y \mid y \text{ est un } n\text{-voisin de } x\} \cap X$
- $N_n^k(x, X) = N_n^{k-1}(x, X) \cup \{y \mid y \text{ 26-voisin de } x \text{ et } \exists z \in N_n^{k-1}(x, X) \text{ tel que } y \ n\text{-voisin de } z\}$

Voisinages géodésiques

Le n-voisinage géodésique de x dans X, noté $G_n(x, X)$ est défini par :
- $G_6(x, X) = N_6^2(x, X)$
- $G_18(x, X) = N_{18}^2(x, X)$
- $G_{26}(x, X) = N_{26}^1(x, X)$

Nombre topologique

Le nombre topologique associé à x et à X est le nombre de composantes n-connexes de $G_n(x, X)$. On le note $T_n(x, X)$.

4.2.2 Caractérisation des points simples

Pour $(n, \overline{n}) \in \{(6, 26), (26, 6)\}$, on a la caractérisation des voxels simples suivante : Le voxel $x \in X$ est n-simple si et seulement si $T_n(x, X) = 1$ et $T_{\overline{n}}(x, \overline{X}) = 1$.

Ce critère est pratique car il est local : pour savoir si un point est simple il suffit de s'interresser à son 26-voisinage.

4.3 Implémentation

4.3.1 Détection des points simples à l'aide de la caractérisation

On recherche les points 26-simples : on doit donc calculer, pour tout voxel x de la scène, $N_{26}^1(x, X)$, $N_6^2(x, \overline{X})$, $T_{26}(x, X)$ et $T_6(x, \overline{X})$. Tout d'abord, pour chaque voxel du 26-voisinage de x, on cherche dans l'octree s'il fait ou non partie de la scène. On remplit ainsi un tableau de 26 booléens qui représente le 26-voisinage de x.
Calcul de $N_{26}^1(x, X)$: à l'aide du tableau de booléens, on construit un graphe G_{26} dont les sommets sont les voxels du 26-voisinage de x qui sont dans la scène. On relie par une arête les sommets dont les voxels correspondants sont 26-voisins.
Calcul de $N_6^2(x, \overline{X})$: à partir du tableau de booléens, on construit un graphe $\overline{G}_6$ dont les sommets sont les voxels du 26-voisinage de x qui ne sont pas dans la scène. Deux sommets sont reliés par une arête si les voxels qu'ils représentent sont 6-voisins. $N_6^2(x, \overline{X})$

est simplement obtenu par un parcours en largeur de profondeur 2 de ce graphe en partant de x.
Calcul de $T_{26}(x, X)$ *et* $T_6(x, \overline{X})$ *:* c'est le calcul du nombre de composantes connexes de G_{26} et $\overline{G}_6$. On applique l'algorithme classique : on fait un parcours en largeur de ces graphes en comptant au fur et à mesure les sommets rencontrés ; si à la fin du parcours, on a rencontré autant de sommets que n'en comptait le graphe, alors il est connexe, sinon il ne l'est pas.

4.3.2 Visibilité d'un point simple

En chaque voxel de la scène, il existe deux normales à l'objet en ce voxel. Mais une seule nous intéresse :

- celle qui est dirigée vers l'interieur de l'objet si la caméra est à l'interieur de l'objet ;
- celle qui est dirigér vers l'exterieur de l'objet si la caméra est hors de l'objet.

Cette normale, notée $\overrightarrow{N}$ est donnée par le logiciel.

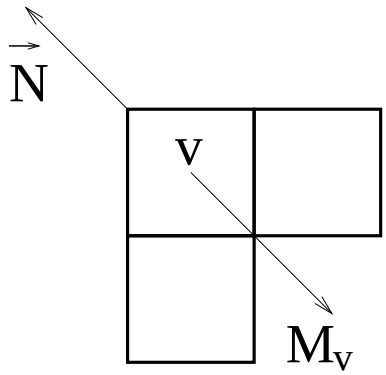


FIG. 4.3 – Un voxel visible

Pour un voxel $v = (i, j, k)$, on définit le point M_v comme le barycentre des voxels voisins de v qui sont dans l'objet. On dit alors que v est visible si $\overrightarrow{N}.\overrightarrow{vM_v} < 0$ (voir fig.4.3).

4.4 Résultats expérimentaux

L'algorithme de suppression de points simple fonctionne convenablement. Cependant, des problèmes provenant d'autres parties du logiciel sont apparus :

- pour beaucoup de voxels, la normale n'est pas définie. Il est donc impossible de savoir si le voxel est visible ou pas. Ce problème n'est pas trop gênant car on peut se contenter de supprimer tous les voxels simples, même ceux qui sont visibles, la différence étant à peine perceptible sur l'image finale.
- les traits noirs dans les angles sont parfois remplacés par de fins liserés. L'origine de ce phénomène n'est pas connue avec certitude et le manque de temps n'a pas

permis de régler ce problème.

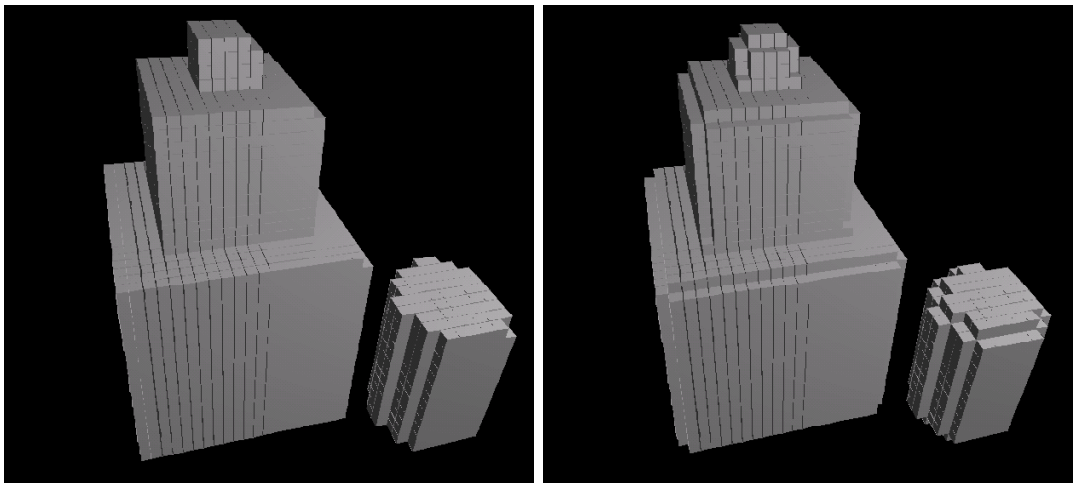


FIG. 4.4 – Suppression des voxels simples

Annexe A

Méthode du lancer de rayons

Le rendu par lancer de rayons est beaucoup plus réaliste que le rendu par l'algorithme du Z-buffer (elle permet les ombres portées, les objets transparents ou réfléchissants), mais nécessite un temps de calcul considérablement plus important.

A.1 Principe

Le lancer de rayon simule la propagation de la lumière. Dans la réalité, les rayons lumineux partent des sources et, après d'éventuelles réflections et réfractions, il est possible qu'ils atteignent l'observateur. Ici, comme on ne s'interresse effectivement qu'aux rayons qui atteignent l'observateur, on ne va calculer que ceux-ci. Pour ce faire, on va simuler la propagation des rayons *à l'envers* : on lance les rayons depuis l'observateur vers les objets de la scène.

On utilise une projection en perspective (modèle de camera *pinhole*) : on a un centre

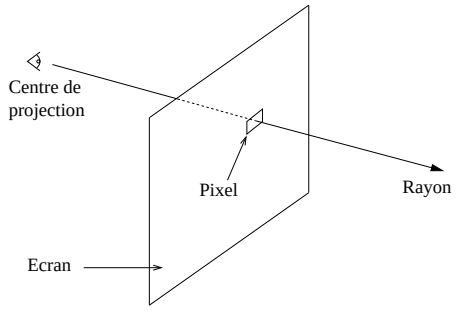


FIG. A.1 – On lance un rayon par pixel de l'écran

de projection, en lequel est placé l'observateur, et un écran (voir fig.A.1). On va alors lancer tous les rayons issus du centre de projection et passant par un pixel de l'écran.

A.2 Calcul des rayons

On considère un rayon partant de l'observateur. Ce rayon rencontre un objet au point P. A partir de ce point P, on va lancer trois types de rayons (voir fig.A.2) :

1. Des *rayons d'éclairement* partants de P et dirigés vers chaque source lumineuse ;
2. Un *rayon réfléchi* si l'objet est réfléchissant ou partiellement réfléchissant ;
3. Un *rayon de transparence* si l'objet est transparent ou partiellement transparent.

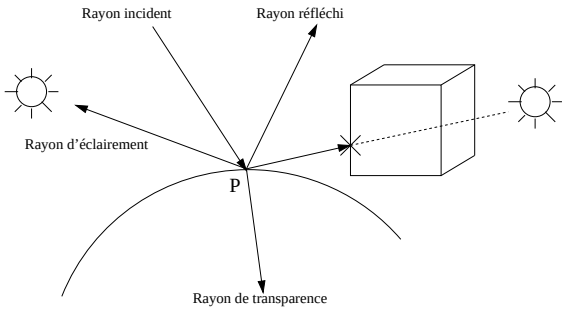


FIG. A.2 – Les differents types de rayons

Le calcul des rayons se prête bien à l'utilisation de procédures récursives : pour calculer l'intensité portée par un rayon, on commence par déterminer le point P, puis on calcule l'influence des rayons d'éclairement (même calcul que pour le modèle du Z-buffer) et ensuite on calcule récursivement l'influence du rayon réfléchi et du rayon de transparence. Il ne reste plus alors qu'à effectuer une moyenne pondérée, dont les coefficients sont caractéristiques de l'objet auquel appartient le point P, des intensités provenant de ces rayons pour avoir l'intensité du rayon incident en P.
On peut arrêter le calcul au bout d'un nombre fixé de réflections/réfractions, ou bien on peut choisir de s'arrêter lorsque la contribution du rayon devient inferieure à une certaine proportion de l'intensité du rayon initial (on parle alors de lancer de rayon adaptatif).

A.3 Les limites du lancer de rayons

Le lancer de rayons est une technique très utilisée pour le rendu d'images. Elle offre de très nombreuses possibilités, surtout en ce qui concerne la transparence et les reflections. Cependant, cette méthode n'est pas exempte de défauts :

- Le calcul de l'image dépend de la position de l'observateur. Ceci rend toute navigation interactive dans la scène impossible car on est obligé de refaire tous les calculs dès que la position de l'observateur est modifiée.
- La réflection diffuse n'est pas gérée : par exemple, si la scène comporte une surface rouge fortement éclairée, les objets qui sont à proximité de cette surface colorée ne seront pas teintés en rouge.

FIG. A.3 – Un exemple de scène obtenue par lancer de rayons

Annexe B

Méthode de la radiosité classique

On rappelle l'équation de base de la radiosité (voir 2.1) :

$$B(x) = E(x) + \rho(x) \int_{y \in scene} B(y) \frac{\cos\theta \cos\theta'}{\pi r^2} V(x,y) dy$$

Résolution de l'équation

La scène étant composée de polyèdres, on considère que la luminosité B, l'emmitance E et le coefficient ρ sont constants sur chaque facette. On note $(F_i)_{i=1..N}$ les facettes de la scène, et B_i, E_i, ρ_i les valeurs de B, E, ρ sur F_i. L'équation devient alors :

$$B_i = E_i + \rho_i \sum_{j=1}^{N} B_j \int_{x \in F_i} \int_{y \in F_j} \frac{\cos\theta \cos\theta'}{\pi r^2} V(x,y) dy dx$$

On appelle facteurs de formes les coefficients $(f_{i,j})$ définis par :

$$f_{i,j} = \int_{x \in F_i} \int_{y \in F_j} \frac{\cos\theta \cos\theta'}{\pi r^2} V(x,y) dy dx$$

On aboutit alors à une forme plus simple de l'équation :

$$B_i = E_i + \rho_i \sum_{j=1}^{N} f_{i,j} B_j$$

Cette équation possède les propriétés nécessaires pour appliquer les méthodes de Jacobi ou de Gauss-Siedel. Ces méthodes sont expliquées au 2.3.2.

Ce qui prend (de loin) le plus de temps dans cette méthode est le calcul des facteurs de forme (integrales doubles sur les facettes). Cependant, ce calcul est indépendant du point de vue. Une fois ce calcul effectué on peut donc naviguer interactivement dans la scène.

FIG. B.1 – Un exemple de scène obtenue par radiosité

Annexe C

Bibliographie

- Rémy Malgouyres, *Algorithmes pour l'informatique graphique*. Dunod, à paraître.
- Rémy Malgouyres, *A dicrete radiosity method*. Proceedings of DGCI'2002, Lecture Notes in Computer Science 2301, Springer, pp 428-438, 2002.
- François X. Sillon et Claude Puech, *Radiosity & Global Illumination*. Morgan Kaufmann Publishers, 1994.
- Sébasien Fourey, *Nombre d'intersection et d'entrelacement de courbes discrètes et application à la caractérisation de la préservation de la topologie en imagerie*, chapitre 10. Thèse soutenue à l'université de Caen, 2000.