SIAMES : SYNTHÈSE D'IMAGES, ANIMATION, MODÉLISATION ET SIMULATION

PROJET SIAMES

RAPPORT DE DEA

Structuration de scènes complexes axiales pour l'optimisation des calculs de visibilité

Florent Coudret

IRISA
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES
Campus de Beaulieu – 35042 Rennes Cedex – France
Tél. : (33) 99 84 71 00 – Fax : (33) 99 84 71 71

Structuration de scènes complexes axiales pour l'optimisation des calculs de visibilité

Florent Coudret *

Projet Siames

Rapport de DEA
Sous la direction de : Eric MAISEL et Kadi BOUATOUCH
Septembre 1995 — 119 pages

Mots-clé : géométrie algorithmique, visibilité, arbre BSP

*. fcoudret@irisa.fr (Stagiaire DEA)

Centre National de la Recherche Scientifique
(URA 227) Université de Rennes 1 – Insa de Rennes

INRIA
Institut National de Recherche en Informatique
et en Automatique – unité de recherche de Rennes

Remerciements

Je remercie M. Bruno Arnaldi, responsable du Projet SIAMES, pour m'avoir accueilli au sein de son équipe.

Je tiens à remercier MM. Eric Maisel et Kadi Bouatouch qui m'ont encadré et soutenu durant ce stage. Je leur suis tout particulièrement reconnaissant d'avoir patiemment étudié et relu les versions préliminaires de ce rapport.

Enfin, je remercie également toute l'équipe du projet SIAMES pour son accueil et sa disponibilité.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Un des objectifs du projet SIAMES concerne la possibilité d'explorer en temps réel des environnements complexes visualisés avec un rendu photo-réaliste. Pour atteindre le photo-réalisme en synthèse d'images, il est nécessaire de simuler correctement l'éclairage des scènes visualisées. La méthode de radiosité répond à cette demande en apportant, en outre, une totale indépendance des calculs par rapport au point d'observation choisi dans une scène. Cette propriété intéresse naturellement l'éclairage d'environnements dans lesquels on puisse se déplacer rapidement, et en particulier, les environnements architecturaux (intérieurs d'immeubles ou artères d'une ville par exemple).

L'algorithme de radiosité exige cependant de "mailler" la surface des objets (c'est-à-dire de la subdiviser en petites surfaces: les carreaux), puis d'établir les relations de visibilité existantes entre les carreaux obtenus. Malheureusement, pour obtenir une bonne précision dans les calculs d'éclairement, un nombre élevé de carreaux est souvent nécessaire, même pour des scènes peu complexes (c'est-à-dire ne contenant qu'un faible nombre d'objets). Cette contrainte augmente considérablement les coûts de calcul et de taille mémoire pour résoudre complètement les échanges d'énergie lumineuse, et limite d'autant la taille des scènes manipulées.

Il est également intéressant de pouvoir traverser et visualiser en temps réel de telles scènes. Le problème principal est d'afficher uniquement les éléments de la scène (par exemple l'ensemble des polygones) effectivement visibles de l'observateur. Pour cela, des calculs de visibilité sont nécessaires pour identifier ces éléments. Malheureusement, la taille des scènes à visualiser est bien souvent trop importante pour pouvoir afficher suffisamment d'images par seconde, diminuant ainsi la fluidité de l'animation. En effet, la limitation vient du nombre de primitives par seconde que les algorithmes de visibilité (même cablés) peuvent traiter.

Pour diminuer ces complexités spatiales et temporelles, ils est possible d'établir certaines informations de visibilité globale sur une scène. Peu de méthodes ont été proposées dans la littérature (celles-ci seront détaillées au chapitre 2). Puisque l'on ne peut prendre en compte l'intégralité d'une scène complexe pour calculer, par exemple, rapidement son éclairement, l'idée est de structurer celle-ci en cellules puis d'établir un graphe de visibilité entre ces cellules. Ainsi, l'éclairage d'une cellule ne nécessite plus la contribution de toute la scène mais uniquement des autres cellules avec lesquelles elle possède des relations de visibilité.

De même, ces relations de visibilité permettent d'accélérer la phase de visualisation d'une scène. Il suffit de limiter l'ensemble des objets potentiellement visibles à ceux présents dans les cellules visibles de la cellule occupée par l'observateur. Ainsi, le nombre de primitives que doivent traiter les algorithmes de visibilité est diminué, ce qui accélère d'autant les temps de traitement et d'affichage d'une image.

Dans les deux cas, il devient possible de traiter des scènes plus importantes en réduisant sensiblement les complexités évoquées plus haut.

Notre approche reprend cette direction en étudiant de manière concrète, et souvent plus approfondie, les facteurs essentiels à une bonne structuration de la scène en cellules. Plusieurs cas particuliers, peu souvent évoqués dans la littérature, sont également traités. Nous proposons ainsi une méthode efficace, basée sur une technique de subdivision binaire de l'espace (Binary Space Partitionning), pour découper une scène en cellules, ainsi qu'un algorithme simple et rapide pour déterminer les visibilités des cellules entre elles.

Ma contribution concerne plus particulièrement la détermination des critères de subdivision conduisant à une structuration "fine" des scènes, ainsi qu'au traitement de certains cas particuliers. De plus, les résultats de ces travaux ne se limitent pas aux calculs de radiosité d'une scène, mais peuvent également être utilisés pour d'autres algorithmes de rendu tels que le Z-buffer ou encore le lancer de rayon.

La suite de ce document est organisée de la manière suivante: le chapitre 2 constitue un état de l'art sur les solutions déjà proposées dans la littérature pour résoudre les problèmes de visibilité intervenant dans la visualisation de scènes complexes. L'idée commune des auteurs repose sur une structuration de la scène en cellules et l'utilisation d'un graphe de visibilité. Les différentes méthodes présentées sont discutées et comparées à la fin de ce chapitre.

Notre contribution est présentée au chapitre 3. Après une brève description du vocabulaire employé et des hypothèses que nous nous imposons, nous nous attachons à décrire notre propre méthode de structuration de la scène en cellules et les critères de découpage retenus. La construction du graphe de visibilité des cellules entre elles est également détaillée: L'algorithme utilisé reste simple et plusieurs améliorations possibles seront signalées. Une description des structures essentielles manipulées (arbre BSP, cellules, graphe de visibilité) termine ce chapitre.

Que ce soit pendant la phase de structuration de la scène ou lors des calculs de visibilité, plusieurs cas non triviaux (ou cas particuliers) se sont révélés. Ces difficultés sont largement développées au chapitre 4.

Plusieurs scènes de complexité croissante nous ont permis d'illustrer les travaux réalisés. Les résultats de structuration et du calcul de visibilité sur ces scènes sont présentés au chapitre 5. Nous discuterons en particulier de l'évolution de ces résultats en fonction des critères de découpage choisis. Il reste cependant évident qu'une structuration "parfaite" n'existe pas, un des objectifs étant qu'elle suive au mieux la topologie de la scène étudiée. Plusieurs statistiques concernant les graphes de visibilité obtenus terminent ce chapitre et nous montrons concrètement comment ces nouvelles informations peuvent être exploitées dans la visualisation interactive de scènes complexes.

Enfin, le chapitre 6 conclut ces travaux en rappelant les contributions apportées à la résolution des problèmes posés initialement. Etant donné les résultats encourageants obtenus, plusieurs voies sont finalement proposées pour poursuivre ce travail.

Chapitre 2

Etat de l'art

Dans ce chapitre, nous présentons différentes méthodes déjà proposées dans la littérature pour diminuer la complexité du calcul de visibilité dans une scène. Une première partie décrit clairement le problème et les traitements nécessaires pour le résoudre. Les travaux déjà effectués dans cette voie sont présentés dans la partie suivante. Finalement, ces méthodes sont discutées dans la dernière partie de ce chapitre.

2.1 Position du problème

Les calculs de visibilité sont primordiaux dans la création d'une image. En effet, pour visualiser correctement une scène, il est nécessaire de déterminer avec précision quelles sont les objets visibles (totalement ou en partie) depuis un point d'observation donné. Dans ce cas, la complexité de ces calculs est liée au nombre d'objets présents dans celle-ci. Ainsi, par exemple, les temps de calcul et la place mémoire nécessaires à la visualisation d'un immeuble tout entier seront naturellement bien supérieurs à ceux demandés par un petit appartement. Mais cette complexité ne dépend pas seulement de la taille d'une scène: si nous choisissons, par exemple, de traverser notre immeuble, l'ensemble des objets visibles peut fortement varier avec notre emplacement (complexité spatiale) ou bien encore selon la dynamique propre à la scène due aux mouvements de certains objets dans celle-ci (complexité temporelle).

Les modèles architecturaux (nos principaux modèles applicatifs) rassemblent facilement ces complexités. Ce sont bien souvent des milieux fortement occlusifs: seule une faible proportion des objets de la scène demeure effectivement visible depuis un point d'observation donné. Ainsi, par exemple, les étages inférieurs (ou supérieurs) à celui occupé par notre observateur lui sont définitivement invisibles, et le nombre d'objets effectivement visibles reste très inférieur au nombre total des objets contenus dans la scène.

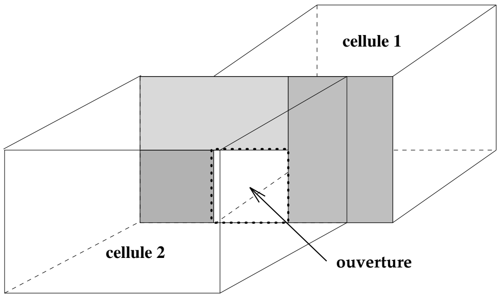

FIG. 2.1 - *Ouverture entre deux cellules*

Les algorithmes de visibilité qui visent, dans ce cas, à réduire le nombre d'objets à afficher, peuvent exploiter cette propriété. Pour cela, il est possible de structurer la scène en cellules et ouvertures. Une cellule est un volume spatial (vide ou peuplé) obtenue par un découpage de la scène. Une ouverture est une région 2D transparente située sur la frontière séparant deux cellules adjacentes. La figure 2.1 montre ainsi deux cellules adjacentes reliées par une ouverture.

Beaucoup de scènes complexes sont naturellement structurées. C'est le cas par exemple des environnements architecturaux (un batiment est constitué d'étages et les étages de pièces). Clairement, le découpage d'un tel modèle s'effectuera de préférence le long des murs, de manière à ce que les cellules correspondent aux pièces de l'immeuble et les ouvertures aux portes et fenêtres. En déterminant les relations de visibilité des cellules entre elles (une cellule n'est visible d'une autre qu'à travers une séquence d'ouvertures), nous accélérons alors fortement la recherche des objets visibles d'un observateur: ils sont nécessairement dans les cellules visibles de la cellule occupée par l'observateur.

De même, les simulations d'éclairage basées sur les échanges d'énergie lumineuse entre les surfaces contenues dans une scène (telle que la radiosité) sont directement concernées par cette structuration de la scène en cellules. Dans la plupart des algorithmes de radiosité, les échanges de rayonnements lumineux sont décrits par l'équation suivante:

$$B(P) = E(P) + \rho_i \sum_{j=1}^{N} \int_{S_j} k(P,P')\, vis(P,P')\, B(P')\, dP' \tag{2.1}$$

où :

- $B(P)$ est l'énergie lumineuse émise par un point P;
- $E(P)$ est l'émission propre de P (cas d'une source lumineuse);
- ρ_i est la réflectivité de la facette i; cette valeur est caractéristique du matériau composant la facette i et représente le rapport de l'énergie réémise sur l'énergie reçue;

- N est le nombre de facettes contenues dans la scène;
- dP' est un élément de surface autour d'un point P';
- $k(P, P')$ désigne la proportion d'énergie émise par P' et reçue par P;
- $vis(P, P')$ vaut 1 si les points P et P' sont visibles et 0 sinon.

Pour une longueur d'onde donnée, l'équation (2.1) indique que la quantité d'énergie lumineuse réfléchie par un point P est égale à la quantité d'énergie émise par P à laquelle s'ajoute une proportion de l'énergie lumineuse incidente.

Dans la plupart des cas, il est impossible de résoudre de façon analytique l'équation (2.1). Des méthodes numériques doivent être utilisées. Pour cela, il est nécessaire de travailler sur une version discrète de l'équation (2.1). Celle-ci est obtenue en subdivisant la scène en petites surfaces : les carreaux. Pour faciliter l'expression de l'équation discrète de la radiosité, on suppose que la subdivision est telle que la radiosité est constante en chaque point d'un même carreau.

La radiosité en un point B_i d'un carreau donné s'exprime par l'équation suivante :

$$B_i = E_i + \rho_i \sum_k B_k F_{ik} \tag{2.2}$$

où :

- l'entier k indice cette fois un ensemble de N carreaux
- ρ_i est la réflectivité du carreau
- F_{ik} est le facteur de forme entre les carreaux i et k c'est-à-dire la proportion d'énergie émise par le carreau k et arrivant sur le carreau i

Pour une bonne approximation de la fonction de radiosité sur ces carreaux, cette subdivision doit être suffisamment fine. Le nombre élevé de carreaux qui en résulte est la cause de la complexité des calculs de radiosité. En effet, l'opération de base de l'algorithme consiste à transférer de l'énergie entre deux carreaux. Il s'agit donc de déterminer si le carreau récepteur est visible ou non du carreau émetteur, et cette étape nécessite d'effectuer des calculs de visibilité sur la scène.

Considérons l'exemple de la figure 2.2. Il s'agit du plan d'un petit appartement (pour simplifier, les murs sont supposés sans épaisseur et les portes ne sont pas représentées). Etant donné un point quelconque dans la pièce F, il est évident que seule une faible partie de la scène contribue à son éclairement. Il est possible de tirer parti de cette propriété pour réduire la complexité des algorithmes : on ne tient compte dans la résolution des radiosités

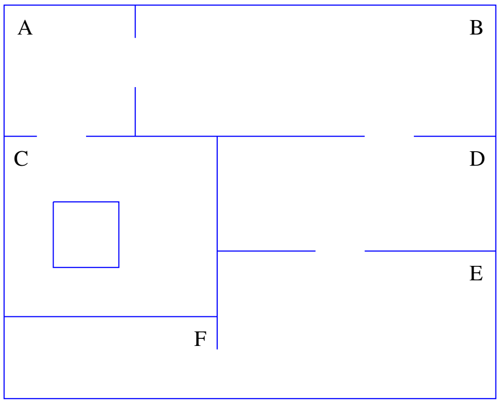

FIG. 2.2 - *Plan d'un appartement*

que des objets susceptibles d'échanger de la lumière avec un point de F.

Le problème revient donc à déterminer l'ensemble des points de la scène visibles depuis la pièce F. Naturellement, les principaux éléments structurants de la scène (tels que les murs, sols, ou plafonds) sont largement responsables des occlusions pouvant limiter cet ensemble. Le problème de la visibilité de la pièce F se ramène à celui de la visibilité d'un observateur mobile dans cette pièce.

Pour traiter ces problèmes, deux étapes apparaissent finalement nécessaires :

1. structurer la scène en cellules (en tenant compte des objets les plus occlusifs);
2. calculer les relations de visibilité entre ces cellules.

Remarquons la similitude entre le problème de l'utilisation du walkthrough et celui de la radiosité. Dans les deux cas, il s'agit d'effectuer des calculs de visibilité entre les points d'une région de l'espace (appartenant à une courbe ou à une surface) et la scène.
La section suivante présentent les travaux déjà effectués dans cette direction.

2.2 Travaux précédents

Différentes approches pour optimiser les calculs de visibilité existent dans la littérature. Par exemple, dans les algorithmes basés sur le lancé de rayons, plusieurs travaux proposent de structurer la scène. Ainsi, dans [1], Bouatouch et al. appliquent une grille spatiale 3D sur la scène. L'espace est régulièrement subdivisé en boites, ou cellules, sans tenir compte des objets qu'il contient. Les algorithmes incrémentaux qui en résultent accélèrent de façon importante le lancer de rayon.

Glasner ([2]) propose une hiérarchie de volumes englobants à base d'octree: l'espace est cette fois subdivisé selon une complexité locale. Cette structure permet d'utiliser des algorithmes récursifs pour le lancé de rayons, et procure une représentation mémoire beaucoup plus compacte de la scène.

Dans [3], Thirion utilise une structure de données particulière, nommée Trie, pour combiner les avantages d'une représentation mémoire compacte (meilleure que celle obtenue avec les octree), avec une rapidité accrue des algorithmes de lancer de rayon.

Mais, ces structurations de la scène sont uniquement faites en fonction de la distribution des objets dans l'espace et non des relations de visibilité. Dans certains cas, on doit tenir compte d'un grand nombre d'objets qui ne seront pas visibles. Nous présentons ci-dessous plusieurs travaux exploitant les relations de visibilité entre objets, pour optimiser les calculs d'élimination d'objets cachés.

2.2.1 Jones : une première approche

Dès 1970, Jones [4] est le premier à décrire un algorithme de visibilité basé sur une structuration de la scène en cellules. Bien qu'à cette époque, les possibilités matérielles soient encore très limitées (seul l'affichage perpective en fil de fer est réalisé sur des écrans de faible résolution), le problème de l'élimination des arêtes cachées est déjà largement étudié. Mais, à la différence des travaux de Jones, les solutions proposées ne permettent pas de visualiser aisément, et de manière (quasi-) interactive, un environnement 3D. En effet, les temps de calculs nécessaires pour éliminer ces “arêtes cachées” sont encore trop importants, et les capacités mémoires bien insuffisantes, pour pouvoir envisager la libre progression d'un observateur virtuel dans une scène quelconque.

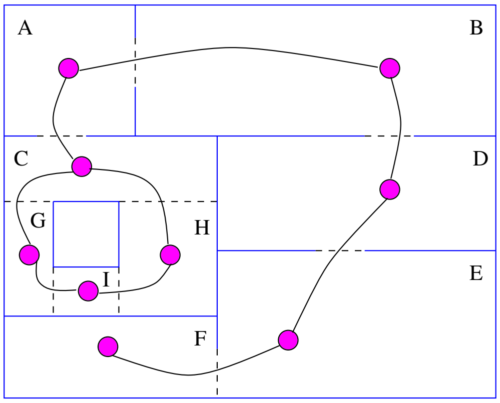

FIG. 2.3 - *Subdivision spatiale 2D et graphe d'adjacence correspondant*

Pour résoudre ce problème, Jones propose de subdiviser la scène en cellules spatiales délimitées par les facettes des objets, d'une part, et par des faces virtuelles transparentes (qu'il appelle ouvertures) d'autre part. Les cellules ainsi établies doivent être convexes. Les ouvertures ainsi introduites sont également convexes (ce sont des polygones) et partagés par deux cellules adjacentes. Une telle décomposition est réalisée par l'utilisateur et peut-être facilement (selon l'auteur) appliquée à un environnement architectural : la figure 2.3 est un exemple de découpage appliqué au plan de notre petit appartement. Les cellules correspondent ainsi quasiment aux pièces et aux ouvertures (représentées en pointillées sur la figure) correspondent pour la plupart aux portes de l'appartement.

La structure représentant l'interconnexion des cellules entre elles est alors un graphe dans lequel chaque noeud représente une cellule et chaque arc une ouverture (cf. figure 2.3). Partant d'une cellule donnée (un noeud du graphe) et d'une direction de visée, une procédure de recherche récursive est alors engagée sur ce graphe. Il s'agit de déterminer à l'affichage quelles cellules sont visibles de l'observateur. Si nous plaçons, par exemple, un observateur dans la cellule A, celui-ci peut voir les murs de sa cellule ainsi qu'une partie de ceux délimitant les cellules B et C au travers des ouvertures. Plusieurs cellules (noeuds dans le graphe) peuvent ainsi être atteintes via une liste d'ouvertures (liste d'arcs). Un exemple d'une telle liste d'ouvertures traversées est présenté figure 2.4.

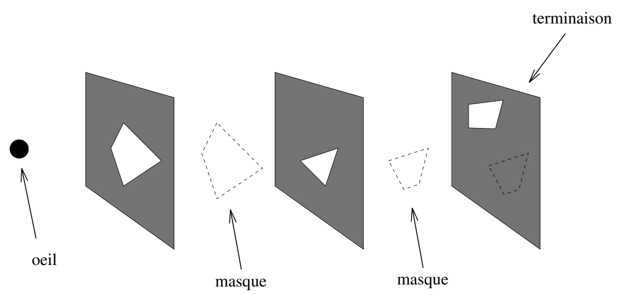

FIG. 2.4 - *Méthode des masques de Jones*

A l'écran, la projection d'une ouverture forme un contour polygonal servant de masque de visibilité. A chaque nouvelle cellule (noeuds du graphe) rencontrée, un nouveau masque de visibilité est calculé. Celui-ci est formé à partir de l'intersection du masque courant et du contour de l'ouverture traversée, projeté sur l'écran. Le masque obtenu permet de déterminer quelles sont les éléments de la cellule atteinte encore visibibles. Ainsi, toutes les lignes situées sur les faces et arêtes intérieures à la cellule (et ne faisant pas partie de l'ouverture traversée) sont projetées sur l'écran et découpées par le masque.

Si l'intersection est vide, l'ouverture est invisible et la progression dans le graphe s'arrête. L'algorithme "remonte" récursivement à la cellule précédente (en retrouvant également le masque précédent) pour détecter d'autres cellules visibles à travers une nouvelle séquence d'ouverture.

Si l'intersection n'est pas vide alors une nouvelle cellule est atteinte et le processus de recherche continue avec examinant ses autres ouvertures.

Notons que l'ensemble des noeuds parcourus (cellules visibles) forment un sous-graphe orienté (cf. figure 2.5) du graphe d'adjacence.

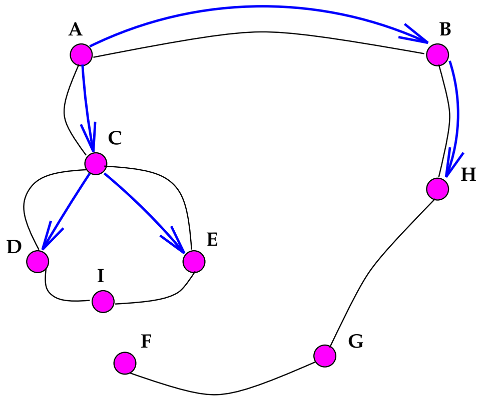

FIG. 2.5 - *Sous-graphe orienté des cellules visibles*

Cette approche a permi à Jones de générer rapidement, et de manière quasi-interactive, plusieurs séquences de vues d'objets 3D ou d'environnements architecturaux (peu complexes néanmoins).

2.2.2 La structuration de scènes automatique d'Airey

Avec la progression spectaculaire de la puissance de calcul et des capacités mémoires disponibles sur les machines, des scènes plus complexes peuvent désormais être traitées et visualisées avec réalisme. Le réalisme d'une image dépend en particulier de la richesse des éléments qui la composent (c'est à dire du nombre d'objets dans la scène) et de la précision avec laquelle ils sont modélisés (par exemple, le nombre de facettes qui les composent) et rendus. L'affichage d'une image est directement pris en charge par un pipeline graphique, qui s'occupe de résoudre les problèmes de visibilité. L'algorithme de Z-buffer est, par exemple, souvent intégré au matériel graphique.

Néanmoins, un goulot d'étranglement subsiste : il s'agit du nombre maximum de primitives géométriques (des facettes par exemple) que le sous-système graphique permet d'afficher (en respectant les visibilités) par seconde. Or simuler la traversée d'un environnement virtuel nécessite de pouvoir afficher plusieurs images de la scène par seconde. A partir d'une fréquence d'environ 20 images par seconde, les saccades dans l'affichage disparaissent (Airey évoque un minimum acceptable de 6 images par seconde).

L'objectif des travaux d'Airey [5] est de garder un haut niveau d'interactivité dans la traversée d'un environnement virtuel (ici, un immeuble) sans toutefois pénaliser le réalisme des images affichées. Pour cela, une stratégie de pré-calcul des relations de visibilité est engagée avant l'étape de la traversée. Ceci permet de diminuer le nombre de primitives que le pipeline graphique doit traiter à chaque mouvement de l'observateur. Airey exploite ainsi plusieurs propriétés caractéristiques des environnements architecturaux:

1. un modèle architectural est moins souvent modifié que les points de vue sur ce dernier, ce qui justifie une étape de précalcul des visibilités avant la traversée;
2. dans la plupart des cas, la majeure partie de la scène ne contribue en rien à une visualisation locale (intérieur d'une pièce dans un immeuble par exemple);
3. beaucoup de polygones sont axiaux (c'est à dire parallèles à deux des trois axes de coordonnées) et rectangulaires;
4. quand un observateur se déplace dans la scène, l'ensemble des polygones visibles varie peu, excepté aux voisinages des ouvertures (portes ou fenêtres par exemple).

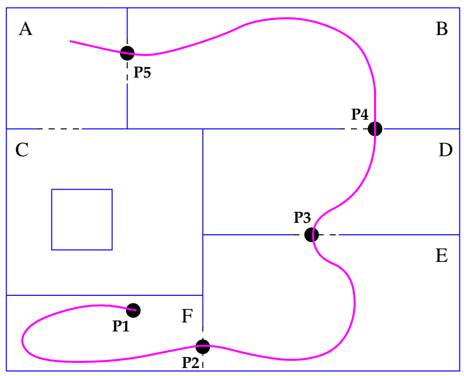

FIG. 2.6 - *Exemple d'itinéraire emprunté par un observateur*

Ce dernier point est illustré par la figure 2.6. La trajectoire possible d'un observateur dans la scène est dessinée. La partie de la scène qui lui est visible depuis le point P1 ne changera certainement pas beaucoup (ce sera principalement les murs et les objets éventuels de la pièce F) jusqu'à ce qu'il se trouve au voisinage du point P2. Son passage par l'ouverture (ici une porte) s'accompagne en effet d'un changement radical de l'ensemble des objets qui lui sont visibles: il voit maintenant surtout les éléments composant la pièce E.

Airey propose donc de subdiviser le modèle en cellules, pour lesquelles l'ensemble des polygones potentiellement visibles (ou encore susceptibles d'être visibles depuis certains points des cellules) est calculé. Comme la taille de ces ensembles est souvent largement inférieure à la taille du modèle tout entier, le processus d'affichage s'en trouve nettement acceléré. Ce découpage de la scène doit cependant suivre deux objectifs :

1. minimiser la taille des ensembles de polygones potentiellement visibles;
2. minimiser le nombre de cellules obtenues (pour éviter en particulier que deux cellules adjacentes ne se retrouvent avec des ensembles de visibilité quasiment identiques).

De plus, l'algorithme de subdivision doit permettre de retrouver aisément la cellule contenant le point de vue courant. Cette contrainte implique l'utilisation d'une structure de données adaptée à une recherche rapide : Airey choisit donc d'utiliser des plans de partition binaire [1] (découpage BSP) et perpendiculaires aux axes du repère (pour exploiter la propriété 3). Finalement, l'algorithme choisit les plans de découpage en fonction des critères suivants:

1. comment le plan découpe de facon équilibrée le modèle;
2. comment le plan cache entre elles les deux régions obtenues (ou encore son degré d'opacité);
3. combien de polygones sont découpés par le plan (un tel polygone sera en effet visible des deux cotés du plan).

Une combinaison de ces critères est effectuée pour obtenir un ordre de priorité parmi les plans de découpage candidats. Le découpage se termine lorque tous les plans candidats ont été choisis. Le processus génère un arbre dont les noeuds représentent les plans de séparation et les feuilles les cellules. Un exemple d'une telle décomposition et de l'arbre obtenu est présenté figure 2.7. Les plans de découpages sont numérotés selon l'ordre de leur sélection.

Une fois la scène structurée en cellules, les sous-ensembles du modèle, visibles d'un observateur mobile dans chacune des cellules, sont calculés et associés à ces cellules. Ces sous-ensembles comprennent les polygones situés sur la frontière et à l'intérieur des cellules obtenues, ainsi que ceux visibles au travers des ouvertures des cellules. Ce dernier groupe de polygones semble pourtant difficile à calculer de manière exacte. Devant la complexité et l'inefficacité des méthodes existantes pour retrouver les polygones visibles depuis une ouverture, Airey opte pour un algorithme calculant de manière approchée cet ensemble: plusieurs rayons de visibilité sont lancés depuis des points aléatoirement choisis sur les ouvertures. Si un polygone est touché par un rayon, il est alors visible (peut-être partiellement) et automatiquement inclus dans l'ensemble des polygones potentiellement visibles.

Les résultats obtenus par Airey montrent des temps d'affichage accélérés d'un facteur variant de 3 à 30. De plus, l'augmentation de la taille mémoire (centrale) nécessaire pour stocker puis gérer ces précalculs de visibilité pendant l'affichage n'excède pas 20%.

1. cette technique sera détaillée plus loin

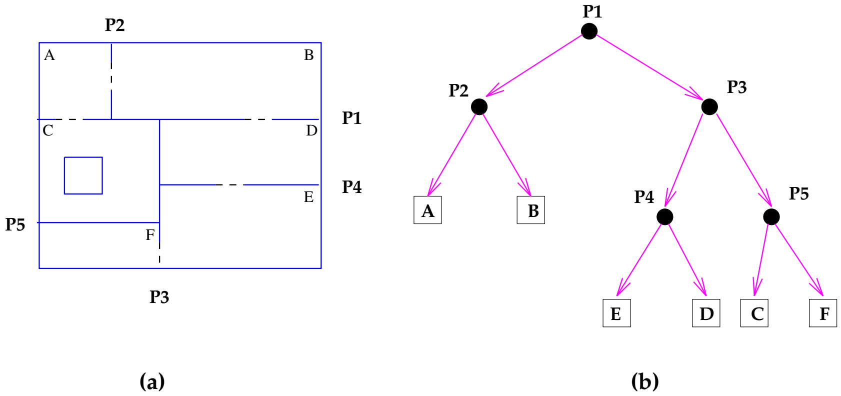

FIG. 2.7 - *Exemple de décomposition d'une scène (a) et arbre BSP obtenu (b)*

2.2.3 Plantiga et les évènements visuels

Le problème de la traversée interactive de modèle 3D complexes (un immeuble par exemple) interesse également Plantiga [6, 7]. Partant des mêmes constatations qu'Airey quant aux difficultés du matériel graphique pour visualiser de telles scènes, il opte également pour une phase de précalculs réduisant sensiblement le nombre de polygones à afficher à chaque image en identifiant ceux qui ne seront pas visibles depuis certaines régions de la scène.

Pour simplifier le problème, Plantiga propose, dans un premier temps, de limiter les mouvements de l'observateur à un espace 2D: le plan parallèle au sol et situé au niveau des yeux d'un promeneur potentiel. L'observateur peut, bien entendu, déplacer son regard autour (et même au-dessous ou au-dessus) de lui, mais il ne peut pas s'élever ou descendre de ce niveau d'observation. La phase de précalculs des relations de visibilité préconisée par l'auteur consiste alors à calculer une partition de ce plan en régions où l'apparence topologique reste constante, c'est à dire depuis lesquelles un même ensemble d'objets est visible. Par objets, Plantiga désigne un ensemble de polygones spatialement compact et non séparés par d'autres faces importantes. Il suppose que la distinction entre les objets et les principaux éléments structurants de la scène (tels que les murs) est établie.

Les objets sont souvent constitués de centaines (voire de milliers) de polygones, et les calculs de visibilité impliquant toutes les faces d'un objet deviendraient très vite excessifs. Pour des raisons d'efficacité, une boite englobante, dont les arêtes sont parallèles aux axes de coordonnées, est calculée pour chaque objet. Les calculs de visibilité sont alors effectués sur cet englobant, plutôt que sur les faces de l'objet. Ainsi, si une boite est invisible depuis un point de vue donné, toutes les faces composant cet objet seront certainement invisibles également. En revanche, la visibilité de la boite implique une visibilité potentielle (et peut-être partielle) de l'objet. Enfin, cette méthode d'approximation de l'objet par une boite englobante autorise la modélisation d'objets par des surfaces courbes sans augmenter la complexité des calculs de visibilité.

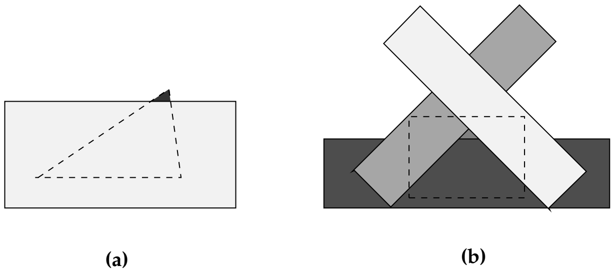

FIG. 2.8 - *Evènements visuels EV (a) et EEE (b)*

L'algorithme identifiant les cellules contenant un même ensemble d'objets potentiellement visibles calcule les évènements visuels: ce sont les points d'observation depuis lesquels l'image change topologiquement s'ils sont traversés. Les évènements visuels apparaissent aux endroits depuis lesquels une arête et un sommet (évènements EV), ou encore trois arêtes (évènements EEE), de la scène semblent s'intersecté. La figure 2.8 présente deux exemples de tels évènements visuels. Dans le premier cas (évènement EV), la facette triangulaire semblera disparaître (ou apparaître) avec le mouvement de l'observateur. Le second cas (évènement EEE) montre que la facette occultée par l'évènement ne contient pas forcément une des arêtes générant l'évènement. Les points d'évènement potentiel forment des lignes de vue coplanaires dans le cas d'évènements EV, et des lignes de vue appartenant à une surface quadrique trouée dans le cas d'évènements EEE. L'intersection de ces surfaces avec le plan d'évolution de l'observateur est un ensemble de segments de droites (pour les évènements EV) et de segments de courbes (pour les évènements EEE) qui partitionnent ce plan en zones 2D dans lesquelles l'ensemble des objets potentiellement visibles ne change pas.

Plantiga utilise finalement un graphe d'évènements visuels pour gérer, durant la traversée, ces ensembles de polygones potentiellement visibles: les arêtes du graphe représentent les segments de droites et de courbes de la partition, alors que les sommets représentent les intersections de ces segments. A chaque sommet, les coordonnées des points d'intersection (dans le plan d'évolution) sont stockées. A chaque arête les constantes représentant la courbe quadratique du segment concerné sont calculées, et la liste des objets à ajouter, ainsi que celle des objets à enlever, sont établies.

Ainsi, lorsque l'observateur se déplace, il devient possible de déterminer les objets qui deviennent invisibles ou potentiellement visibles en vérifiant s'il quitte la cellule courante. Dans ce cas, il suffit d'effectuer une simple mise à jour de la liste des objets à afficher en fonction des listes d'ajout et de retrait présentes sur l'arête du graphe représentant le segment de frontière traversé.

Cette approche a permis à Plantiga de visualiser la traversée de grosses scènes pouvant notamment contenir des surfaces courbes. La partition de l'espace des points de vue (le plan d'observation) en cellules est automatiquement générée à partir d'une sélection des objets et des murs faites sur la scène. Les ensembles de polygones potentiellement visibles précalculés sont conservatifs (i.e. contiennent au minimum l'ensemble des polygones effectivement visibles) et c'est le pipeline graphique qui calcul en dernier ressort les visibilités exacts.

2.2.4 Les travaux de Teller

Les différentes publications de Teller [8, 9, 10, 11] proposent également de structurer les scènes architecturales pour traiter des problèmes de visibilité. Cependant, les algorithmes présentés s'adressent non seulement au problème de leur traversée interactive, mais également aux calculs d'illumination globale de telles scènes (comme la méthode de radiosité par exemple).

Constatant la difficulté à déterminer exactement l'ensemble des objets visibles depuis un point d'observation donné, Teller choisit plutôt de précalculer un surensemble de ces objets. Cette première phase entraine la structuration de la scène en cellules spatiales convexes séparées par des murs. Tout comme Plantiga, il suppose que la représentation du modèle distingue clairement les murs (ou principaux éléments occlusifs de la scène) des autres objets.

La décomposition utilise un arbre BSP (Binary Space Partitionning) et se base (comme le découpage d'Airey) sur le degré d'opacité des polygones dans la scène (cf. définition plus loin). L'algorithme part d'une cellule initiale correspondant à la boite englobante de tous les polygones de la scène, exceptés ceux décrivant les objets. Un plan de découpage candidat est alors celui dont l'aire totale des polygones supportés dépasse une certaine fraction de l'aire du polygone résultant de l'intersection du plan avec la cellule courante. La figure 2.9 illustre la présence de tels plan de découpage potentiels dans une cellule. La section s1 (respectivement s2) est le polygone résultant de l'intersection entre la cellule et le plan supportant le polygone P1 (respectivement P2). Le plan supportant le polygone P1 (resp. P2) est alors candidat comme plan de découpage si l'aire de P1 dépasse une certaine fraction (par exemple 30 %) de l'aire de s1 (resp. s2).

Une fois la décomposition en cellules achevée, les ouvertures situées sur les frontières des cellules sont calculées. Finalement, la subdivision spatiale est représentée par un graphe d'adjacence sur ces cellules tel celui présenté figure 2.3. Deux cellules sont ainsi reliées s'il

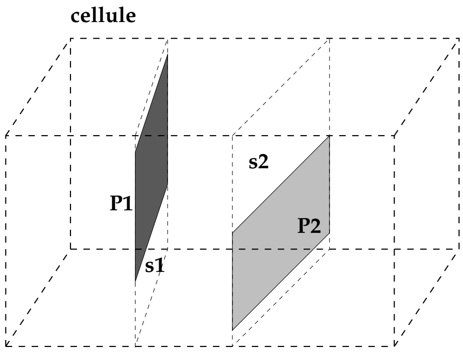

FIG. 2.9 - *Plans de découpage d'une cellule*

existe une ouverture entre elles.

A partir du graphe d'adjacence des cellules, Teller calcule alors pour chaque cellule l'ensembles des cellules qui lui sont visibles. Cette opération nécessite de parcourir récursivement le graphe d'adjacence (partant d'un noeud donné) en déterminant à chaque nouveau noeud traversé, si la cellule correspondante est visible (souvent partiellement) ou non de la cellule de départ. Une cellule ne peut "voir" ses voisines qu'à travers ses ouvertures, et les cellules plus éloignées qu'au travers d'une séquence d'ouvertures. Ainsi, s'il existe un rayon traversant cette séquence d'ouvertures sans rencontrer d'obstacle alors les cellules situées aux deux extrémités sont visibles. La figure 2.10 montre l'existence d'un tel rayon traversant une séquence d'ouvertures.

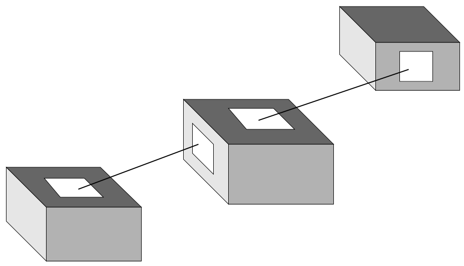

FIG. 2.10 - *Rayon de visibilité traversant une liste d'ouvertures*

Dès lors que la visibilité entre deux cellules est établie, Teller construit les volumes effectivement visibles dans chacune des cellules. En effet, si deux cellules sont visibles à travers une séquence d'ouvertures, seule une portion de l'une d'elle est visible de l'autre (et réciproquement). Cette dernière remarque est illustrée par la figure 2.11. Les cellules B, D et E sont les cellules atteintes pendant la recherche des cellules visibles depuis la cellule F. Mais seules les zones grisées de ces cellules seront visibles d'un observateur circulant dans la cellule F. Connaissant ces volumes de visibilité, il est alors facile d'identifier les objets potentiellement visibles depuis une cellule: ce sont ceux qui intersectent ces mêmes volumes.

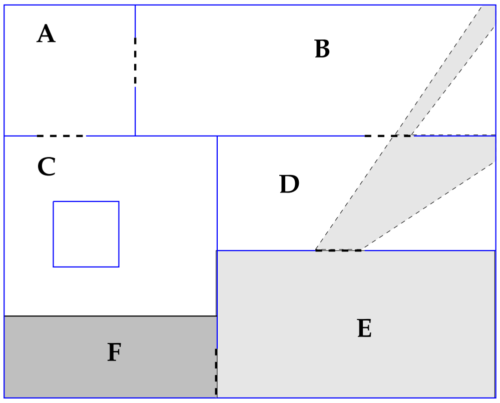

FIG. 2.11 - *Regions visibles depuis une cellule*

Tous ces calculs correspondent à une étape de prétraitement des calculs de visibilité d'une scène donnée. Durant la phase dynamique (déplacement d'un observateur dans la scène), l'algorithme de visualisation détermine la cellule dans laquelle se trouve l'observateur. Connaissant cette cellule, les cellules (et même les parties de ces cellules) potentiellement visibles de celui-ci sont automatiquement retrouvées. L'ensemble des polygones que doit traiter le pipeline graphique se trouve donc largement réduit puisqu'il ne contient plus que les polygones qui appartiennent aux parties des cellules visibles (murs et objets), et non pas tous les polygones de la scène.

Pour accélérer les calculs d'illumination globale (et notamment le calcul des radiosités) dans de tels environnements, Teller propose de pousser un peu plus loin la phase de précalcul des visibilités. Rappelons que la méthode de radiosité repose sur un découpage des surfaces de la scène en carreaux. Pour calculer correctement les échanges d'énergie lumineuse entre ces carreaux, il est nécessaire de savoir si des occlusions existent entre eux, c'est à dire d'effectuer des calculs de visibilité sur la scène.

Pour réduire le coût de ces calculs, Teller identifie tout d'abord les paires de polygones mutuellement visibles dans la scène. Cette opération s'effectue en partant des informations de visibilité inter-cellules déjà calculées. Pour chaque polygone d'une cellule, l'auteur calcule le volume de la scène qui lui est visible en traversant les séquences d'ouvertures établies lors du calcul des visibilités inter-cellules. Quand une cellule est atteinte (en progressant dans le graphe), seuls les polygones de la cellule situés dans ce volume peuvent être visibles. Si la cellule atteinte est vide (i.e. dépourvue d'objets), ces polygones sont entièrement visibles du polygone de départ. Dans le cas contraire, certains polygones peuvent être partiellement ou complètement cachés par les objets. Finalement, cette phase de précalcul des visibilités produit pour chaque polygone P de la scène un ensemble de polygones visibles V(P). Puis, pour chaque polygone R appartenant à V(P), une liste de bloqueurs B(P,R), contenant tous les polygones cachant potentiellement la visibilité des polygones P et R, est construite. C'est cette dernière liste qui est exploitée puis mise à jour dynamiquement pendant la phase de calculs des échanges lumineux entre les polygones de la scène.

En effet, dans l'algorithme de radiosité hiérarchique [12] (utilisé par Teller), quand l'échange lumineux entre deux carreaux ne peut être établi sous un certain seuil d'erreur global, un des deux carreaux de l'interaction est automatiquement subdivisé en carreaux plus petits. Cette erreur est causée par l'existence d'occlusions (partielles) dans la visibilité des carreaux entre eux. Les occlusions sont elles-même la conséquence de la présence de bloqueurs (d'autres carreaux de la scène) entre les carreaux. Les carreaux fils issus du carreau subdivisé peuvent également échanger de l'énergie avec l'autre carreau. Il convient donc à nouveau de connaître les bloqueurs éventuels de ces nouvelles interactions. Ces nouveaux ensembles de bloqueurs sont nécessairement des sous-ensembles des bloqueurs existants entre les carreaux initiaux.

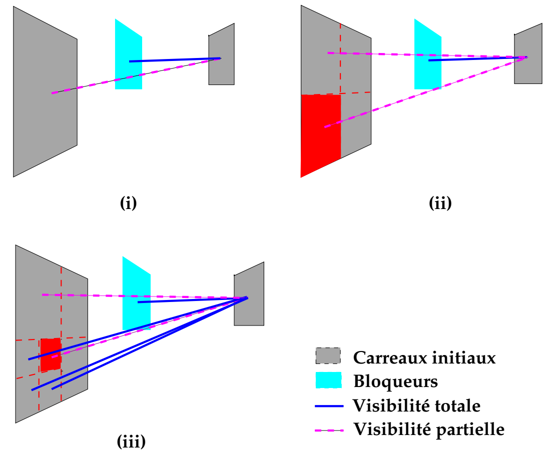

FIG. 2.12 - *Reclassification des interactions après subdivision*

La figure 2.12 présente trois étapes de reclassification des interactions entre carreaux après subdivision. La partie (i) montre l'interaction initiale considérée à savoir deux carreaux dont la visibilité mutuelle est partiellement cachée par un bloqueur. Comme l'erreur causée par ce bloqueur dans leurs échanges lumineux est trop importante, l'un des deux (ici, celui de gauche) est subdivisé en carreaux plus petits (partie (ii)). La visibilité résultante entre l'un des carreaux fils et l'autre carreau est encore partielle, et l'erreur commise serait encore trop grande. Une deuxième subdivision (partie (iii)) augmente finalement le nombre de liens de visibilité totale.

Les algorithmes présentés par Teller pour établir toutes ces informations de visibilité (visibilité inter-cellules, inter-polygones, listes de bloqueurs) ont permis d'optimiser sensiblement les subdivisions nécessaires pour établir correctement les transferts d'énergie lumineuse entre les carreaux. Ses travaux permettent ainsi un calcul plus rapide des radiosités dans un environnement polygonal complexe.

2.2.5 La mise en oeuvre de Luebke

L'algorithme de Teller ne parait pas suffisament efficace à Luebke dans le cadre d'une visualisation interactive d'un environnement complexe. Dans [13], Luebke et al. décrivent une méthode simple pour évaluer rapidement les polygones potentiellement visibles lors de la visualisation dynamique d'un modèle architectural. Leur approche est similaire à celle de Jones dans le sens où, partant d'une structuration de la scène en cellules et ouvertures, ils déterminent seulement à l'étape de visualisation les cellules visibles depuis le point d'observation.

La décomposition de la scène en cellules et ouvertures est directement intégrée à la phase de modélisation de la scène. Une cellule est donc représentée par un regroupement de polygones, et n'est plus forcément convexe. Les ouvertures correspondent à des polygones particuliers qui contiennent (en plus des informations géométriques les décrivant) le nom des cellules ainsi connectées. Enfin, les objets modélisés appartiennent automatiquement à une cellule de la scène.

Pour évaluer dynamiquement l'ensemble des polygones potentiellement visibles d'un point, les auteurs travaillent directement sur leur projection (et selon la direction de visée de l'observateur) dans l'espace écran. Partant d'une cellule donnée (la cellule contenant l'observateur), chacune de ses ouvertures est ainsi projetée sur le plan image. La projection d'une ouverture sur l'écran donne un polygone dont une boite englobante 2D peut être calculée. Le rectangle obtenu, appelé rectangle de visibilité, sert à identifier les objets des cellules voisines qui ne seront pas visibles: ce sont ceux dont la projection (de leur englobant) sur le plan image tombe en dehors du rectangle. A chaque ouverture traversée, un nouveau rectangle de visibilité est calculé en prenant l'intersection du rectangle de visibilité courant avec celui résultant de la projection de l'ouverture sur l'écran. La visibilité des objets de la nouvelle cellule atteinte est donc testée avec ce nouveau rectangle de visibilité. Comme un même objet peut être visible à travers différentes séquences d'ouvertures, les objets sont marqués dès qu'ils sont affichés à l'écran, évitant ainsi de les réafficher plusieurs fois.

Cet algorithme de visibilité a permi d'écarter 20 à 50 % des polygones d'une scène pour visualiser sa traversée. Même si ces résultats dépendent étroitement du modèle et de la trajectoire empruntée pendant sa visualisation, l'algorithme présenté reste néanmoins une technique d'accélération simple et efficace.

2.3 Discussion

A une époque où le Z-buffer câblé n'existait pas encore, les travaux de Jones ont permi de jeter les bases d'une nouvelle approche permettant des calculs de visibilité à partir d'une structuration de la scène.

Malheureusement, l'étape du découpage de la scène en cellules et la construction du graphe d'adjacence sont des opérations qu'il réalisait manuellement, ce qui limitait considérablement la taille des scènes qu'il était possible de traiter (en dehors des limitations évidentes du matériel informatique de l'époque). En effet, un architecte, ou toute personne modélisant un objet, n'a sûrement ni le temps, ni peut-être même la compétence, pour retravailler, à la main, les bases de données graphiques de son modèle en vue d'améliorer un algorithme d'affichage.

De plus, l'algorithme proposé par Jones implique que toutes les faces d'une scène appartiennent aux frontières des cellules, limitant certainement la méthode aux scènes vides, c'est à dire constituées uniquement des éléments structurants (murs, plafonds) et dépourvues d'objets (chaises, tables). En fait, la distinction entre les objets et les principaux éléments occlusifs n'est pas faite, et la méthode n'est applicable qu'à de simples environnements sans dynamique propre ou sans possibilité de déplacer interactivement des objets.

Aujourd'hui, certains modeleurs (tels ceux utilisés par Luebke et al.) structurent directement les scènes en cellules et ouvertures. L'avantage est que la subdivision en cellules obtenue suit de près la topologie de la scène. Les cellules correspondent effectivement aux pièces d'un immeuble et les ouvertures marquent l'emplacement des portes et des fenêtres. Mais, outre le fait que ces modeleurs soient encore peu nombreux, leur portée se limite à accélérer la visualisation des scènes pendant leur modélisation. Luebke et al. ne traitent ainsi pas du problème de visibilité dans les calculs d'illumination globale d'une scène.

De plus, la hiérarchie imposée par le modeleur (et choisie par l'architecte) sur les polygones d'une scène (structuration en objets, cellules, ouvertures) n'est pas toujours adaptée pour visualiser efficacement certaines scènes. Par exemple, la répartition des polygones de la scène vers les cellules définies par l'architecte est rarement homogène: les corridors ou les halls présents dans certains immeubles contiennent certainement plus de polygones et d'ouvertures que les différentes pièces attenantes. Ce dernier point explique l'extrême variabilité des résultats, obtenus par Luebke et al., dans le pourcentage de polygones définitivement écartés pour visualiser une scène le long d'une trajectoire donnée.

Finalement, Luebke et al. ne font aucun calcul de visibilité globale entre les polygones, ou même les objets, présents dans la scène. Leur algorithme exploite uniquement la structuration de la scène déjà réalisée par le modeleur, et laisse encore beaucoup trop de polygones à traiter par le pipeline graphique. L'approche permet seulement d'augmenter l'interactivité en permettant de visualiser rapidement les modifications apportées dans la modélisation d'environnements architecturaux.

Airey introduit une combinaison de plusieurs critères pour choisir les plans de découpage qui structureront la scène en cellules. Le critère d'opacité compte néanmoins pour beaucoup dans ce choix (il sera d'ailleurs repris par Teller). Mais il ne précise pas quel seuil minimum doit être choisi (si un tel seuil existe !) pour limiter le nombre de cellules obtenues. En effet, l'augmentation du nombre de cellules entraine également celui des ouvertures identifiées sur leurs frontières, ce qui ralentit les calculs de visibilité inter-cellules. De plus, les ensembles de polygones potentiellement visibles depuis une cellule deviennent ainsi peu différents de ceux des cellules voisines.

La difficulté, dans l'approche d'Airey, est bien de calculer ces ensembles de polygones potentiellement visibles (ou PVS) pour une cellule donnée. Le problème se réduit en fait à la détermination des polygones visibles depuis les ouvertures de cette cellule. L'algorithme proposé par Airey calcule une approximation de cet ensemble, et peut dans certains cas sous-estimer l'ensemble des polygones effectivement visibles. En effet, si nous considérons la figure 2.13, nous voyons bien que certains objets de la scène, pourtant visibles depuis l'ouverture, échappent aux rayons de visibilité lancés. Ce résultat est du à l'échantillonnage aléatoire de l'ouverture par un nombre fini de points, et depuis lesquels sont lancés un nombre fini de rayons.

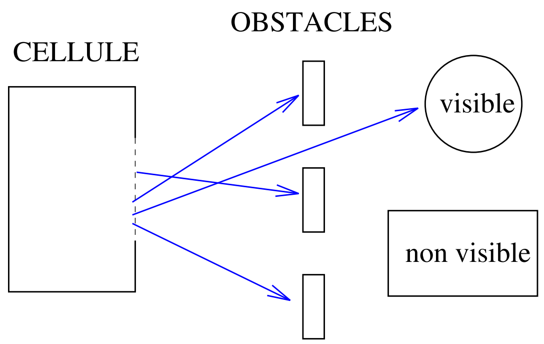

FIG. 2.13 - *Rayons de visibilité aléatoires depuis une ouverture*

Teller évite ce problème en calculant plutôt un sur-ensemble de ce PVS. Il commence par calculer les relations de visibilité inter-cellules en déterminant l'existence ou non d'un rayon (plutôt que de le calculer) traversant la liste des ouvertures menant d'une cellule à l'autre. Notons que pour une séquence d'ouvertures comportant n arêtes, cet algorithme opère en un temps $O(n)$, alors que l'algorithme de Jones (qui, lui, calcule une visibilité exacte pour affichage) s'exécute en un temps $O(n \log n)$. Une fois établie ces relations de visibilité inter-cellules, il précalcule également les relations de visibilité cellule-objets. Pour ce faire, il détermine une solution analytique au problème de la visibilité à travers une séquence d'ouvertures.

Cette approche reste néanmoins mathématiquement complexe et nécessite plusieurs heures de précalculs sur des scènes importantes ([9]). De ce fait, elle n'est pas applicable à la visualisation d'environnements pendant leur modélisation, le calcul du graphe de visibilité n'étant pas incrémental. Sur ce point, la mise en oeuvre de Luebke et al. semble donc plus appropriée que celle de Teller.

Finalement, les différentes méthodes présentées montrent les points communs suivants:

1. calcul d'un ensemble de polygones potentiellement visibles (pre-clipping);
2. structuration de la scène selon des relations de visibilité;
3. utilisation d'un graphe de visibilité dont les sommets sont des régions dont les contenus sont délimités par des éléments occlusifs de la scène.

Le graphe de visibilité obtenu permet ainsi de réduire la complexité des algorithmes de visibilité pour visualiser la traversée d'environnements complexes d'une part, et pour les calculs d'illumination globale d'autre part. Au chapitre suivant, nous présentons notre approche en gardant à l'esprit cette polyvalence d'application.

Chapitre 3

Notre approche

Les différentes méthodes exposées au chapitre précédent se basent toutes sur une même idée: structurer la scène pour réduire la complexité du calcul des visibilités. Notre approche reprend cette idée en approfondissant en particulier l'étape de structuration. En effet, les travaux précédents ne traitent pas, par exemple, des facteurs conduisant à une structuration efficace de la scène, c'est à dire permettant d'obtenir des calculs de visibilité performants. De plus, nous avons choisi de ne pas limiter les résultats de notre méthode à la traversée d'environnements. Nous préférons ainsi obtenir des informations de visibilité globale qui puissent également servir aux calculs d'éclairement d'une scène.

Ce chapitre est organisé comme suit: la section 3.1 décrit globalement notre algorithme en insistant notamment sur les hypothèses prises sur les scènes manipulées. Quelques définitions des principaux termes utilisés dans la suite de ce rapport sont également introduites. La section 3.2 s'intéresse à l'étape de structuration proprement dite de la scène en cellules et ouvertures. Nous nous attachons, notamment, à expliquer l'intérêt d'utiliser un arbre BSP dans ce but. Le graphe de visibilité inter-cellules est obtenu en parcourant récursivement un graphe de visibilité adjacente sur ces cellules. La construction de ces deux graphes est exposée section 3.3. Les principales structures manipulées (arbre BSP, cellules, graphe d'adjacence et de visibilité), ainsi que les différents fichiers générés par notre programme, sont finalement détaillés section 3.4.

3.1 Notre algorithme

Le calcul effectué est le suivant: une scène axiale est lue, puis après traitement, un ensemble de fichiers décrivant le graphe de visibilité est écrit.

Après avoir présenté les étapes principales de notre algorithme (section 3.1.1), quelques restrictions concernant les scènes traitées seront exposées (section 3.1.2). Enfin, nous expliquerons quelques termes couramment employés dans la suite de ce rapport (section 3.1.3).

3.1.1 Les principales étapes

Globalement, notre algorithme se décompose en 4 étapes:

1. Lecture et préparation de la scène;
2. Structuration en cellules et ouvertures;
3. Construction du graphe de visibilité;
4. Ecriture des fichiers résultats.

La première étape concerne la lecture du fichier décrivant la scène. Notons que celle-ci doit être entièrement lue avant de commencer l'étape de structuration. En effet, les plans successifs choisis pour découper la scène en cellules sont (quasiment) tous portés par des polygones de la scène. La sélection de tels plans nécessite une analyse portant sur tous les polygones, et, pour ne pas exagérément ralentir cette opération, l'ensemble de ces polygones est stocké en mémoire centrale. Finalement, ce stockage suit une disposition particulière permettant facilement d'obtenir les plans de découpage (cette préparation est décrite section 3.2.3).

Le découpage de la scène en cellules et ouvertures est effectué à l'étape 2. Cette étape est largement décrite section 3.2.

La troisième étape de notre algorithme est d'établir un graphe de visibilité inter-cellules, c'est à dire d'établir pour chacune des cellules l'ensemble des cellules potentiellement visibles. La construction du graphe de visibilité est exposée section 3.3.

Pour terminer, un certain nombre de fichiers résultats sont produits à l'étape 4. Ils concernent, en particulier, la structuration de la scène et les visibilités inter-cellules obtenues. Le détail de ces fichiers, ainsi que celui des principales structures manipulées par notre programme, est exposé section 3.4.

Pour étudier l'influence des critères de découpage retenus sur la structuration de la scène, un mode statistique est également mis en oeuvre dans notre programme. Celui-ci mesure plusieurs paramètres concernant notamment l'arbre BSP, les cellules et les ouvertures obtenues. Dans ce cas, l'étape 3 (calculs des visibilités inter-cellules) n'est pas réalisée pour des raisons de temps de calcul, et les fichiers résultats (produits à l'étape 4) concernent uniquement l'évolution de ces paramètres.

3.1.2 Hypothèses de travail

Avant de commencer à décrire l'algorithme, un certain nombre d'hypothèses doivent être faites. Les premières vont nous permettrent de traiter des scènes générales:

- les scènes traitées par notre algorithme sont uniquement constituées de facettes planes (tous les modeleurs permettent de représenter les scènes au moyen de facettes);
- aucune structure logique ou hiérarchique entre ces facettes n'existe dans les fichiers décrivant ces scènes (pas de regroupements de facettes en objets par exemple);
- aucune information ne permet d'aider à la structuration. Cela permet de traiter aussi bien les scènes créées au moyen d'un modeleur spécialisé, que celles obtenues sur les sites externes.

Le format d'entrée choisi pour ces fichiers est le format NFF. Une description de ce format est donnée en annexe.

Les secondes vont nous permettrent de simplifier l'algorithme et ne pas trop alourdir les routines de calculs. D'autre part, ces simplifications vont nous permettre de focaliser l'attention sur certains points particuliers, à savoir le choix des critères de structuration de la scène ainsi que le traitement de certans cas particuliers:

- nous nous limitons aux scènes quasi-axiales et composées principalement de facettes axiales. Une scène quasi-axiale est une scène dont les facettes sont pratiquement toutes perpendiculaires à un des axes du repère de la scène. Un exemple de tels facettes axiales est montré figure 3.1. Cette hypothèse n'est pas trop restrictive dans la mesure où les principaux plans structurants dans les scènes architecturales (tels que les murs, sols et plafonds) sont très souvent axiaux.
- ces plans structurants sont, pour la plupart, modélisés sous forme de facettes rectangulaires. Cette seconde hypothèse nous permet de réduire sensiblement la complexité de détection des ouvertures sur les frontières des cellules. En effet, les plans de découpage étant composés de facettes rectangulaires, les ouvertures seront aussi rectangulaires et obtenues simplement par soustraction de facettes rectangulaires entre elles.

Notons que ces deux restrictions ne nous empêchent pas de traiter des scènes comportant des facettes non axiales ou non rectangulaires. En effet, dans un premier temps, seules les facettes axiales et rectangulaires sont retenues pour déterminer les plans de découpage et construire les cellules. A l'issue de ce découpage, les facettes non encore considérées sont replacées dans ces cellules. Ces dernières facettes composent souvent les objets (au sens distincts des principaux éléments structurants) d'une scène.

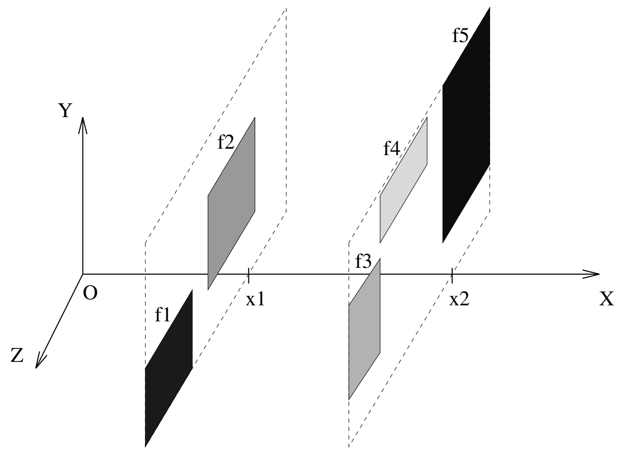

FIG. 3.1 - *Exemple de facettes axiales*

3.1.3 Définitions

Nous donnons, dans cette section, la définition de plusieurs termes couramment employés dans la suite de ce rapport. Ces définitions sont indispensables pour une bonne compréhension des algorithmes présentés.

Nous appelons **scène** l'ensemble des facettes lues dans un fichier. Notons qu'aucune information sémantique ne permet de distinguer une facette d'une autre. Seules des critères géométriques (par exemple, la surface) permettent de le faire.

Toutefois, il est possible de distinguer deux types de facettes selon leur influence sur les relations de visibilité dans la scène. Il s'agit des facettes occlusives et des facettes d'objets plus petits. Les premières sont des facettes qui cachent un grand nombre de points situés de part et d'autre d'elles. Par exemple, dans le cas d'une scène d'intérieur, un mur est une facette occlusive, par contre une lampe ne l'est pas.

L'étape de structuration découpe cette scène en **cellules**. Une cellule est une région convexe de la scène dont les frontières sont constituées d'objets occlusifs. Si la cellule est vide (c'est à dire non peuplée par des objets) alors tous les points de cette région sont nécessairement mutuellement visibles.

Une **facette murale** est une facette située sur la frontière d'une cellule, c'est à dire sur une des faces de sa boite englobante. La figure 3.11 section 3.2.5.2 nous montrera comment détecter les ouvertures d'une cellule à partir de ses facettes murales.

Un **plan axial** est un plan parallèle à deux des trois axes du repère. Les plans de découpage utilisés par notre algorithme sont tous des plans axiaux. Ces plans contiennent des facettes axiales.

Une **facette axiale** est une facette perpendiculaire à un des axes du repère de la scène. Un exemple de facettes axiales est montré figure 3.1.

Une **boite axiale** est une boite dont les faces sont axiales. Les cellules issues de notre structuration seront toutes des boites axiales.

Une **position axiale** est la position d'une facette axiale ou d'un plan axial sur l'axe du repère parallèle à sa normale. Sur la figure 3.1, la position axiale x1 (resp. x2) des facettes f1 et f2 (resp. f3, f4 et f5) est obtenue par l'intersection du plan axial supporttant ces facettes et de l'axe parallèle à leur normale.

Une **liste axiale** est une liste de plans ou de polygones axiaux dont les normales sont toutes parallèles au même axe du repère, et triée selon les positions axiales. La liste axiale correspondante aux facettes de la figure 3.1 serait ainsi *(x1(f1,f2), x2(f3, f4, f5))*.

Nous introduisons également le **degré d'opacité d'un plan par rapport à une cellule.** Cette grandeur nous sera utile lors de la construction des cellules d'une scène. Il s'agit du rapport entre l'aire totale des facettes de la cellule présentes sur ce plan, et l'aire de la section de la cellule à la position axiale du plan. Sur la figure 3.2, le degré d'opacité du plan P par rapport à la cellule est le rapport entre l'aire de la facette f et l'aire de la section s (en pointillé) de la cellule.

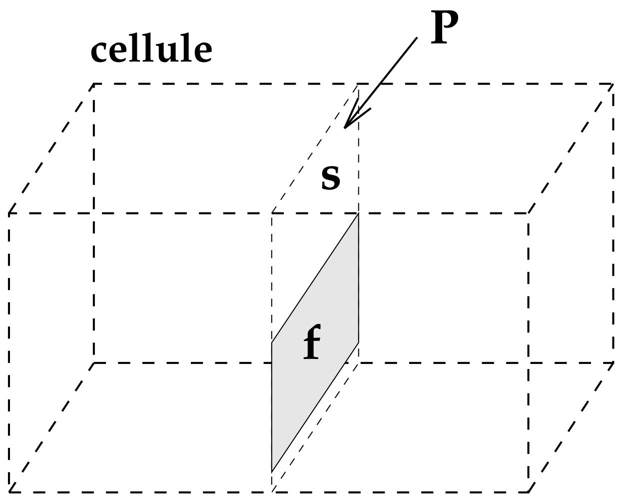

FIG. 3.2 - *Degré d'opacité d'un plan par rapport à une cellule*

Enfin, l'**aspect d'une cellule** désigne le rapport des longueurs de deux cotés orthogonaux d'une cellule. Pour obtenir un rapport inférieur ou égal à 1, la longueur la plus petite est au numérateur. Une cellule possède donc 3 aspects différents selon les cotés mesurés. L'aspect d'une cellule permet d'apprécier sa forme dans les 3 directions de l'espace: plus un aspect se rapproche de 0, plus la cellule est allongée (dans la direction du coté mesuré au dénominateur). La figure 3.3 illustre l'évolution de l'aspect d'une cellule selon une direction donnée.

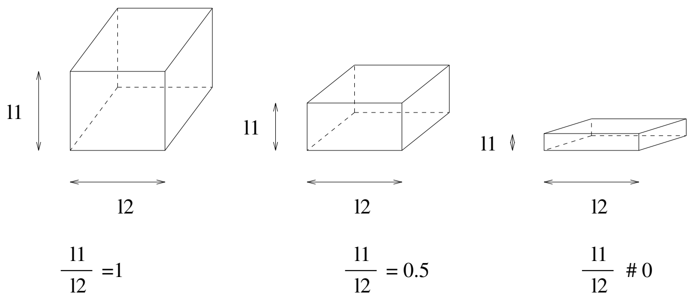

FIG. 3.3 - *Variation de l'aspect d'une cellule*

3.2 Structuration de la scène

Afin de construire un graphe de visibilité, il faut structurer la scène en cellules et ouvertures. Pour cela, nous avons choisi une technique basée sur les arbres BSP. La description des arbres BSP ainsi que les raisons de ce choix sont tout d'abord exposées section 3.2.1. Plusieurs critères permettant de sélectionner un plan de découpage sont ensuite présentés section 3.2.2. Pour tenir compte des hypothèses prises sur la scène (section 3.1.2), et ainsi faciliter le choix des plans de découpage, une disposition particulière de la scène en mémoire est nécessaire. Cette préparation est décrite section 3.2.3. L'algorithme de découpage proprement dit est détaillé section 3.2.4. Nous avons vu que les cellules ne sont (partiellement) visibles entre elles qu'à travers une liste d'ouvertures. Il est donc également nécessaire de calculer toutes les ouvertures présentes dans la scène. Ces calculs sont développés section 3.2.5.

3.2.1 Les arbres BSP

Plusieurs méthodes de structuration d'un espace existent déjà dans la littérature (par exemple, diagramme de Voronoi, triangulation, etc...). Les arbres BSP représentent une structuration d'une scène en cellules convexes, basée sur une décomposition binaire récursive de l'espace.

3.2.1.1 Origine et intérêts

Les arbres BSP (Binary Space Partitioning) ont été introduits pour la première fois par Schumaker et al. dans [14]. Il s'agissait pour l'époque d'utiliser ces structures de données pour effectuer les calculs d'élimination de surfaces cachées dans les simulateurs de vol. PLus tard, Naylor et al. [15] utilisent les arbres BSP dans la séparation linéaire de polygones plans pour modéliser des objets solides.

Le premier intérêt des arbres BSP réside dans l'affichage d'une scène. Couplés à l'algorithme du peintre par exemple (affichage des facettes de la plus éloignée à la plus proche), ils permettent d'obtenir plusieurs vues différentes d'une même scène fixe. Leur utilisation dans la construction d'arbre CSG (Constructive Solid Geometric) et dans les techniques de lancer de rayon est également présentée par Naylor et al. dans [16]. Enfin, Chin [17] s'en sert pour calculer des ombres portées.

La portée des arbres BSP semble donc inépuisable et de nombreuses améliorations concernant leur construction ont depuis été proposées dans la littérature [18, 19, 3, 20].

3.2.1.2 Définition d'un arbre BSP

La définition d'un arbre BSP et son principe de construction peuvent être formellement décrits par la figure 3.4 et les étapes suivantes:

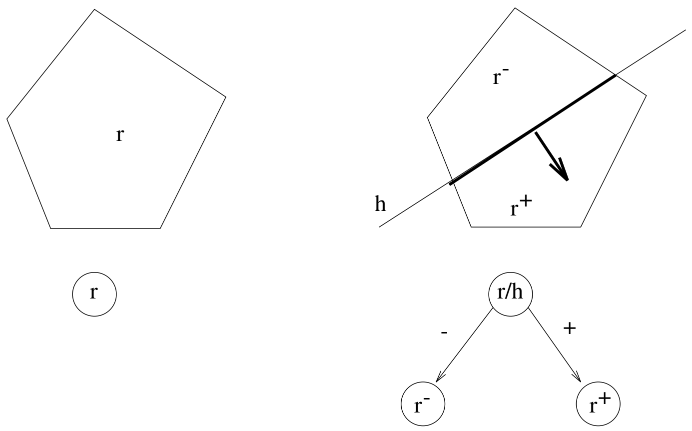

FIG. 3.4 - *Principe de construction d'un arbre BSP*

- soit une région r de dimension d;
- soit un hyperplan h (choisit selon un certain critère) intersectant r;

- alors h partitionne r en 2 sous-régions : $r^+ = r \cap h^+$ et $r^- = r \cap h^-$ où h^+ et h^- sont les 1/2 plans positifs et négatifs (selon la normale à h) de r;
- on peut récursivement recommencer le processus sur r^+ et r^-.

Dans le cas où $d = 2$ (resp. $d = 3$), les hyperplans sont des droites (resp. plans) et les cellules obtenues des polygones (resp. polyhèdres). Un arbre binaire de régions est ainsi construit. Quand le processus est arrêté (selon un critère d'arrêt donné), les feuilles de l'arbres sont appelées cellules. Par construction, il s'agit de régions convexes.

Dans la suite de ce rapport, nous examinerons uniquement le cas où $d = 3$ (pour des raisons pratiques, la plupart des figures illustrent cependant des cas où $d = 2$).

3.2.1.3 Arbre BSP contraint / non contraint

Le choix des hyperplans peut suivre ou non les plans de support des polygones d'une scène. Dans le premier cas, on parle d'arbre BSP contraint. L'ensemble des arbres BSP contraints pour une scène donnée est fini. Les arbres BSP non contraints autorise aussi des plans de découpage correspondants à des polygones virtuels. Dans ce cas, l'ensemble des arbres BSP possible est infini.

La construction d'un arbre BSP peut être incrémentale. Une construction incrémentale signifie que les polygones plans sont ajoutés dynamiquement à la structure.
Cette démarche ne peut être retenue dans le cas où la sélection du plan de découpage dépend de l'ensemble des polygones contenus dans la scène.

Les critères de découpage retenus sont introduits dans la section suivante.

3.2.2 Les critères de découpage

Notre étude vise à structurer la scène en cellules ayant un ensemble de visibilité réduit. Dans le cas des scènes architecturales, il existe déjà de larges plans limitant de façon importante la visibilité. Ce sont par exemple les murs, les sols et les plafonds dans un immeuble.

L'objectif est donc de retrouver ces plans de découpage potentiels dans la scène. Pour cela, nous avons retenu le **critère d'opacité** comme critère principal pour retenir les plans de découpage potentiels. Un plan de découpage potentiel est un plan dont le degré d'opacité sur la cellule à découper est supérieur à un seuil donné. La définition du degré d'opacité d'un plan est donnée section 3.1.3.

En appliquant ce critère, plusieurs plans peuvent être candidats, mais un seul doit être retenu. Il convient alors d'appliquer un second critère (qualifié de secondaire) pour le sélectionner. Nous avons introduit trois critères secondaires différents pour comparer leurs effets sur la structuration de la scène:

- choix du **plan médian**: ce critère sera détaillé section 3.2.4;
- choix du **plan d'opacité maximum**: sélection du plan présentant le plus grand degré d'opacité dans la cellule courante;
- choix du **plan coupant le moins de facettes**: sélection du plan découpant le moins de facettes dans la cellule courante.

Le découpage s'arrête quand le critère principal (degré d'opacité) ne donne plus de plans candidats. La section 3.2.4.2 détaille plus largement la mise en oeuvre de ces critères pour découper une cellule. Les cellules obtenues peuvent, cependant, être encore découpées. De nouveaux critères, basés notamment sur le volume et l'aspect des cellules obtenues, seront introduits au chapitre 4.

3.2.3 Préparation de la scène

D'après les hypothèses de travail évoquées section 3.1.2, seules les facettes axiales et rectangulaires doivent être considérées pour structurer la scène. A la lecture du fichier, les autres facettes sont donc insérées dans une liste d'attente. Une fois la structuration en cellules établie, ces facettes seront redistribuées automatiquement sur leurs cellules respectives.

Pour démarrer l'algorithme BSP, nous devons établir une cellule initiale englobant toute la scène. Comme nos plans de découpage sont tous axiaux, en prenant une boite axiale, toutes les cellules suivantes seront également des boites axiales.

Afin d'obtenir rapidement le degré d'opacité des plans potentiels pour découper une cellule, l'aire des facettes est calculée à leur lecture. Les facettes sont ensuite réparties, selon leur axe, dans les trois listes axiales du repère. Chaque élément de ces listes représente une position axiale. Pour chaque position axiale, nous trouvons donc les facettes partageant cette position et leur aire totale.

3.2.4 La structuration en cellules

La structuration de la scène revient à regrouper des facettes entre elles dans des cellules. Ce regroupement nécessite de découper la scène par des plans axiaux.

Partant d'une cellule regroupant toutes les facettes de la scène, un algorithme découpe récursivement cette cellule selon les plans sélectionnés par les critères de la section 3.2.2. L'arbre de découpage obtenu est un arbre BSP.

L'étape sélectionnant un plan de découpage dans une cellule est présentée section 3.2.4.1.

La section 3.2.4.2 décrit le découpage de la cellule proprement dit.

Une fois les cellules définitives obtenues, nous devons calculer leurs facettes murales (voir définition section 3.1.3); ces calculs sont exposés section 3.2.4.3.

Finalement, les facettes non axiales et non rectangulaires viennent peupler les cellules. Cette dernière étape est présentée section 3.2.4.4.

3.2.4.1 La sélection du plan de découpage

Partant d'une cellule initiale contenant les facettes dans les trois listes axiales (voir 3.2.3), il s'agit de déterminer un plan découpant cette cellule en deux autres cellules contenant à leur tour trois listes axiales. Le critère principal (critère d'opacité) nous permet d'établir une liste de plans potentiels pour ce découpage. Ce sont tous les plans dont le degré d'opacité dans la cellule sont supérieurs ou égaux à un seuil donné. De plus, ces plans potententiels sont répartis selon leur normale dans 3 listes axiales. Plusieurs cas sont envisageables:

1. Les trois listes axiales de la cellule sont vides:

   Dans ce cas, la cellule ne contenant plus de facettes, le découpage sur critère d'opacité s'interrompt.

2. Les trois listes axiales des plans potentiels sont vides:

   Cette fois-ci, il reste encore des facettes dans la cellule courante mais les plans les supportant ne répondent plus au critère d'opacité. Le découpage selon ce critère s'arrête également.

3. Il existe au moins une liste axiale de plans potentiels non vide:

   Si une seule de ces listes est non vide et qu'un unique plan la compose, ce plan est automatiquement retenu pour découper la cellule. Dans le cas contraire, le critère secondaire permet de sélectionner un seul plan.

Considérons le troisième cas où plusieurs plans potentiels sont retenus par le critère d'opacité. Le problème est de choisir un seul de ces plans pour découper la cellule courante en deux nouvelles cellules. Malheureusement, il n'existe pas de fonction indiquant si tel plan est meilleur que tel autre. C'est pourquoi, nous introduisons trois critères secondaires différents pour comparer à postériori les structurations obtenues:

- **critère du plan médian:**

  L'algorithme de découpage sur la scène utilise un arbre BSP. Chaque noeud de l'arbre représente un plan de découpage et les feuilles obtenues sont les cellules finales de la structuration. Pour pouvoir rapidement localiser dans quelle cellule se trouve un point de la scène, il suffit alors de descendre récursivement cet arbre, en comparant la position du point par rapport aux plans rencontrés sur les noeuds. Le nombre de comparaisons nécessaires peut fortement varier d'un point à un autre si l'arbre est mal équilibré. Pour cette raison, il peut être intéressant de vouloir sélectionner des plans de découpage de manière à équilibrer au maximum l'arbre BSP final: c'est l'objectif du choix du plan médian. Des 3 listes de plans potentiels, nous ne retenons que la plus longue, et nous choisissons le plan le plus median de cette liste, pour répartir équitablement les autres plans dans chacune des cellules obtenues.

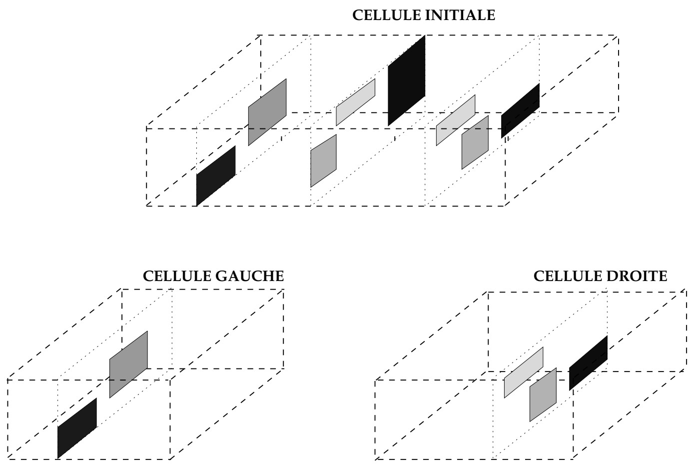

FIG. 3.5 - *Découpage d'une cellule par plan médian*

La figure 3.5 illustre le découpage d'une cellule contenant une seule liste axiale de plans potentiels (représentés en pointillés). La cellule initiale est découpée en deux cellules (gauche et droite) contenant chacune d'entre elles les autres plans potentiels. Ces cellules pourront elles-même être découpées selon ces plans.

- **critère du plan d'opacité maximum:**

  L'objectif de la structuration de la scène est de réduire la complexité du calcul des visibilités. Au chapitre 2, nous avons vu que cette complexité est notamment liée à la taille des ensembles d'objets potentiellement visibles depuis une cellule. L'idée est donc de réduire au maximum la visibilité des cellules à l'étape du découpage. Pour cela, il suffit de choisir le plan de la cellule à découper dont l'opacité est maximum. C'est ce que fait le critère du choix du plan d'opacité maximum. Si plusieurs plans ont la même opacité, alors le critère du plan médian est appliqué. Les plans prioritaires pour découper un immeuble par exemple, sont ainsi les sols, les plafonds et les murs. Dans ce cas, les cellules obtenues suivent plus naturellement les étages et les pièces.

- **critère du plan découpant le moins de facettes:**

La troisième alternative pour choisir le plan de découpage répond au problème du nombre de facettes découpées par l'arbre BSP. Plus le nombre de coupures engendrés par les plans est élevé, plus la taille de la scène au final sera importante. Même si l'on ne peut prédire à l'avance le nombre de nouvelles facettes créées, nous pouvons toujours minimiser ce nombre: il suffit de choisir le plan coupant le moins de facettes dans la cellule courante. Si plusieurs plans ont un nombre de coupures minimal, alors le critère du plan médian est appliqué. Notons que ce critère produit des cellules sensiblement égales au critère d'opacité maximum en réduisant toutefois leur nombre.

3.2.4.2 Le découpage d'une cellule

Une fois le plan de découpage établi, la cellule courante est découpée en deux nouvelles cellules (les cellules filles). Les facettes de la cellule courante sont redistribuées (selon leur position par rapport au plan de découpage) vers ces nouvelles cellules. Certaines nécessitent d'être découpées: ce sont les facettes coupées par le plan de découpage retenu. Finalement, trois nouvelles listes axiales sont établies dans chacunes des deux cellules filles.

Seules les facettes présentes sur le plan de découpage ne sont pas réparties dans les cellules filles. Elles sont conservées, pour traitement ultérieur (voir section 3.2.4.3), dans le noeud de l'arbre BSP correspondant au plan de découpage.

Le découpage reprend alors récursivement sur les deux nouvelles cellules créées.

3.2.4.3 Calcul des facettes murales

Quand le découpage est terminé, nous obtenons alors au niveau des feuilles de l'arbre BSP des cellules pour la plupart encore vides. En effet, seules quelques facettes axiales non retenues par les critères de découpage peuvent rester dans ces cellules. Mais les facettes murales (voir définition section 3.1.3) de chaque cellule ne sont pas encore établies. Ces facettes se trouvent encore au niveau de chaque noeud de l'arbre BSP construit, et doivent être "descendues" dans leur cellule respective. En parcourant récursivement l'arbre BSP, nous récupérons ces facettes sur une boite englobant chaque cellule obtenue. Notons que les facettes présentes à un noeud de l'arbre n'appartiennent pas forcément à la même cellule. En effet, ces facettes peuvent encore être découpées par un plan perpendiculaire situé sur autre noeud plus bas dans l'arborescence. De plus, il convient de respecter le sens d'orientation des normales de ces facettes: la facette murale d'une cellule a une normale orientée vers l'intérieur de cette cellule.

3.2.4.4 Insertion des facettes non traitées

Les facettes non sélectionnées par l'algorithme sont insérées dans les cellules obtenues.

Les facettes non axiales et non rectangulaires doivent également être réparties dans leur cellule respective. A l'issue du découpage, ces facettes sont donc insérées dans les cellules en comparant leur position par rapport aux plans de découpage rencontrés dans la descente de l'arbre BSP. Notons que certaines de ces facettes peuvent également être découpées par les plans rencontrés.

3.2.5 La détection des ouvertures

Les ouvertures se situent sur la frontière séparant deux cellules voisines. Dans un premier temps, il est donc nécessaire d'établir le graphe de voisinage des cellules (section 3.2.5.1). Une fois ce graphe établie, les ouvertures peuvent être calculées (section 3.2.5.2).

3.2.5.1 Le graphe de voisinage

Le graphe de voisinage permet d'établir les relations de voisinages entre les cellules:

- un sommet du graphe représente une cellule;
- deux sommets sont liés si les deux cellules correspondantes ont une frontière commune.

La frontière séparant deux cellules voisines est l'intersection de leur côté voisin. La forme d'une cellule étant une boite axiale, elle possède alors six côtés sur lesquels il faut chercher des cellules voisines. Chaque côté d'une cellule est inclus dans un plan de découpage présent dans l'arbre BSP. Pour retrouver les cellules voisines sur un côté de la cellule, il faut donc retrouver le noeud de l'arbre BSP contenant ce plan de découpage. Finalement, les cellules voisines sont à rechercher en descendant le sous-arbre ne contenant pas la cellule de départ.

La procédure de recherche exige donc de pouvoir à la fois monter et descendre dans l'arbre BSP. Les liens ascendants sont établis en même temps que les liens descendants pendant la construction de l'arbre. Ainsi, à partir d'un noeud de l'arbre, il est possible de remonter au noeud père ou de descendre sur l'un de ses deux noeuds fils.

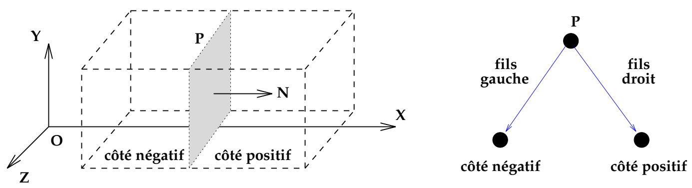

FIG. 3.6 - *Conventions de découpage d'une cellule*

Dès qu'un plan de découpage (noeud de l'arbre) contenant un côté de la cellule est trouvé, il faut redescendre dans le bon sous-arbre, c'est à dire celui ne contenant pas la cellule de départ. Pour ne pas se tromper, des conventions ont été prises pendant la construction de l'arbre BSP, elles sont illustrées par la figure 3.6:

1. Les plans de découpage introduits ont tous une normale orientée dans le sens positif des axes. Sur la figure, le plan de découpage P de la cellule a une normale N orientée dans le sens croissant des X.

2. Le fils gauche d'un noeud correspond toujours au côté négatif du plan de découpage, et le fils droit au côté positif (côté de la normale). Sur la figure, l'arbre BSP du découpage de la cellule est représenté.

Connaissant la face de la cellule retrouvée sur un noeud, il est ainsi facile de redescendre du bon côté pour trouver les cellules voisines.

Lors de cette descente, le noeud rencontré peut-être terminal ou non:

Cas d'une feuille: La cellule atteinte est une cellule voisine de la cellule initiale. Un lien de voisinage est alors créé entre ces deux cellules.

Cas d'un noeud: Dans ce cas, nous arrivons encore sur un plan de découpage. Il s'agit maintenant de déterminer dans quels fils descendre pour retrouver le ou les cellules voisines. Tout dépend en fait de l'orientation de ce plan de découpage par rapport au dernier plan de découpage rencontré. Deux cas sont à envisager:

- Les deux plans sont parallèles.

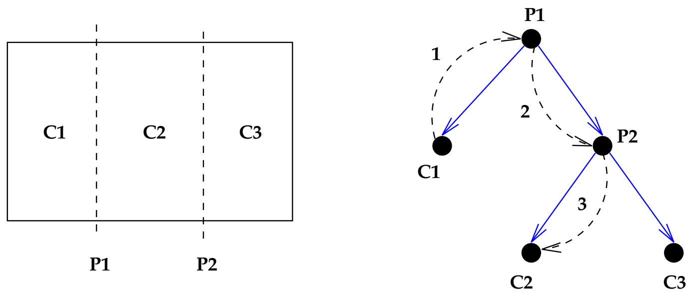

FIG. 3.7 - *Premier type de voisinage entre cellules*

Cette situation est illustrée par la figure 3.7. Celle-ci représente les cellules obtenues (notées C1, C2 et C3) après le découpage d'une scène selon les plans P1 et P2, et l'arbre BSP correspondant. Nous voulons retrouver les cellules voisines de la cellule C1. La procédure de recherche d'un côté de la cellule aboutit au noeud contenant le plan P1 (étape 1 sur la figure). En redescendant l'arbre, nous arrivons sur le noeud du plan P2. Ce plan étant parallèle au plan P1, un seul des deux noeuds fils doit être traversé. Il s'agit du noeud correspondant au côté situé entre les plans P1 et P2. L'étape 3 nous amène donc sur la cellule C2 qui est alors enregistrée comme étant voisine de C1.

– Les deux plans sont orthogonaux.

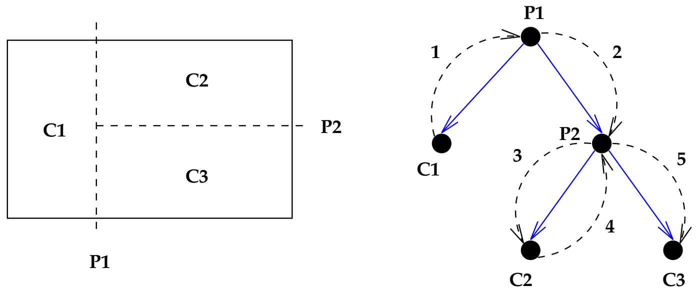

FIG. 3.8 - *Second type de voisinage entre cellules*

Cette situation est illustrée par la figure 3.8. Les mêmes étapes (étapes 1 et 2) que dans le premier cas nous ont conduits jusqu'au noeud du plan P2. Cette fois pourtant, le plan P2 est perpendiculaire au plan P1. Dans ce cas, les deux noeuds fils doivent être parcourus puisqu'aussi bien C2 et C3 sont voisins de C1. Le parcours de notre arbre suit donc les étapes 3, 4 et 5 indiquées sur la figure. Les cellules voisines trouvées sont bien les cellules C2 et C3.

Une exception existe cependant à cette rêgle: si le deuxième plan coupe le premier plan en dehors des limites du côté de la cellule, un seul noeud fils doit être parcouru. Pour mettre en évidence cette exception, considérons la figure 3.9. Il s'agit toujours d'établir les cellules voisines de la cellule C1. Les étapes 1 à 5 de la recherche dans l'arbre suivent les deux rêgles évoquées ci-dessus. Arrivé au noeud du plan P3, nous nous trouvons dans le cas où les deux derniers plans rencontrés (P1 et P3) sont perpendiculaires. Il serait cependant faux de descendre sur les deux noeuds fils de l'arbre. En effet, le plan P3 sortant des limites du côté de la cellule examinée, seule la cellule C4 est voisine de C1.

Lorsque nous rencontrons un plan perpendiculaire au précédent nous devrons donc également tester s'il intersecte ou non la cellule du départ.

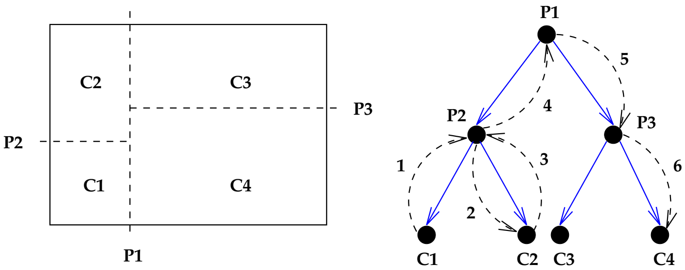

FIG. 3.9 - *Cas particulier de voisinage entre cellules*

Finalement, une cellule ayant 6 côtés, il faut aussi rechercher les 6 noeuds des plans correspondants en remontant dans l'arbre. L'algorithme déterminant les cellules voisines d'une cellule initiale est donc le suivant:

CHERCHERVOISIN(noeud)
Début
    répéter
        répéter
            noeud = *père*(noeud)
        jusqu'à noeud = plan d'un côté
        selon côté trouvé
            noeud = *fils_gauche*(noeud)
            ou noeud = *fils_droit*(noeud)
        si noeud = cellule **alors**
            ajouter cellule à V
        sinon si *plan*(noeud) // plan côté alors
            selon côté trouvé
                CHERCHERVOISIN(*fils_gauche*(noeud))
                ou CHERCHERVOISIN(*fils_droit*(noeud))
        sinon si*plan*(noeud) intersecte cellule **alors**
            CHERCHERVOISIN(*fils_gauche*(noeud))
            et CHERCHERVOISIN(*fils_droit*(noeud))
    jusqu'à avoir traiter les 6 côtés
Fin

La figure 3.10 représente une scène d'intérieur. Le graphe de voisinage des cellules est également dessiné. Notons que le nombre de cellules voisines peut fortement varier d'une cellule à l'autre. Ainsi, la cellule C5 à 9 cellules voisines, alors que la cellule C1 n'en a que 2.

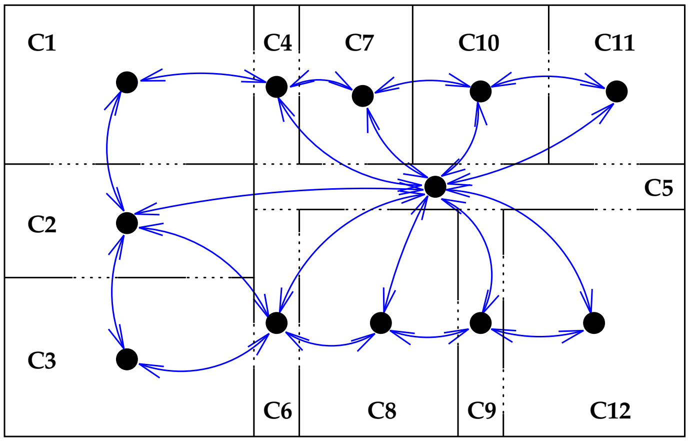

FIG. 3.10 - *Graphe de voisinage*

3.2.5.2 Calculer les ouvertures

A partir du graphe de voisinage, nous pouvons maintenant calculer les ouvertures des cellules. Deux cellules voisines (reliées par un arc dans le graphe) ont une frontière commune. Cette frontière est un polygone rectangulaire puisqu'il est le résultat de l'intersection des deux côtés voisins des cellules. Les facettes murales des deux cellules sont présentes sur cette frontière. Les ouvertures correspondent alors aux zones vides sur la frontière, c'est à dire non recouvert par des facettes murales.

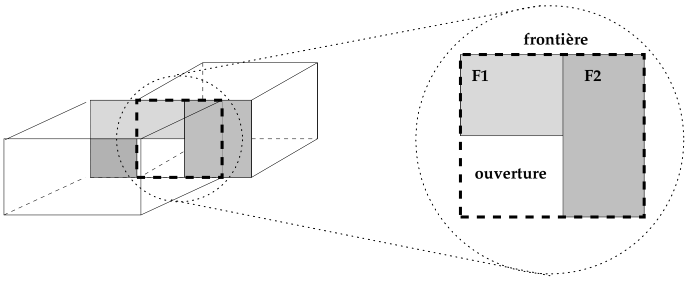

FIG. 3.11 - *Frontière séparant deux cellules voisines*

La figure 3.11 montre ainsi deux cellules voisines. Elles partagent la frontière 2D rectangulaire indiquée en gros pointillés. Pour calculer les ouvertures, nous devons soustraire une par une les facettes murales de cette frontière, à savoir F1 et F2 sur la figure. Comme les facettes murales sont rectangulaires (voir section 3.1.2, nous pouvons facilement obtenir des ouvertures rectangulaires.

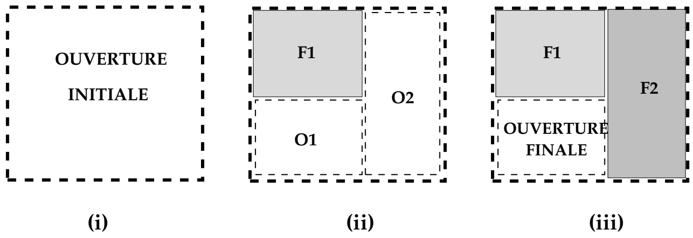

FIG. 3.12 - *Calcul des ouvertures entre deux cellules voisines*

La démarche est illustrée figure 3.12. Nous partons d'une ouverture initiale correspondant à toute la frontière commune des deux cellules (partie (i) de la figure). En prenant une première facette murale et par une opération de soustraction avec l'ouverture initiale, nous obtenons une liste d'ouvertures rectangulaires (O1 et O2 sur la partie (ii)). On reprend ensuite une deuxième facette murale et on recommence la soustraction sur toutes les ouvertures restantes. Quand toutes les facettes murales ont été soustraites, la liste des ouvertures restantes nous donne les ouvertures effectives entre les deux cellules (partie (iii) de la figure). Notons finalement que plusieurs ouvertures peuvent exister entre deux cellules voisines.

3.3 La visibilité inter-cellules

Le graphe de visibilité permet d'établir les relations de visibilité inter-cellules: chaque noeud du graphe désigne une cellule, et un arc entre deux noeuds signifie que les deux cellules correspondantes sont mutuellement visibles (souvent en partie). Les cellules sont visibles entre elles au travers d'une séquence d'ouvertures. A partir du graphe d'adjacence défini section 3.3.1, nous construirons ces séquences d'ouvertures section 3.3.2 pour déterminer le graphe de visibilité.

3.3.1 Le graphe d'adjacence

Considérons la visibilité de deux cellules entre-elles. Celle-ci n'est possible qu'à travers une séquence d'ouvertures. Partant de la première cellule, nous traversons une ouverture qui nous mène à une seconde cellule, et ainsi de suite jusqu'à la dernière cellule.

Pour une cellule donnée, nous devons donc connaître les cellules voisines qui lui sont accessibles par une ouverture. Le graphe d'adjacence est construit en même temps que les ouvertures sont calculées (voir section 3.2.5.2):

- un sommet du graphe représente une cellule;

– un arc entre deux noeuds représente une ouverture permettant de passer d'une cellule à l'autre. Il existe donc autant d'arcs partant d'un noeud que d'ouvertures calculées sur la frontière de la cellule correspondante.

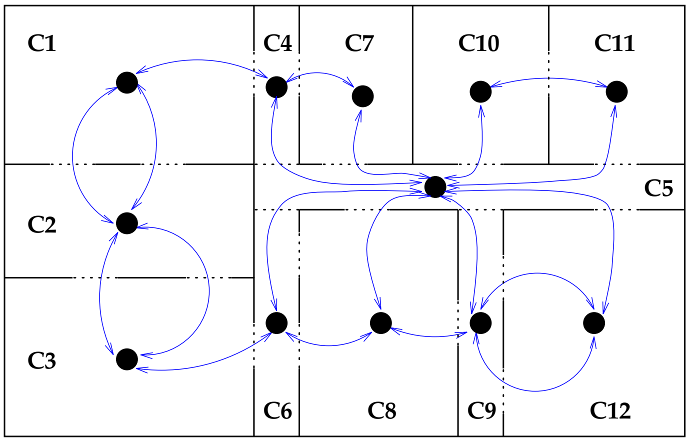

FIG. 3.13 - *Graphe d'adjacence*

La figure 3.13 représente le graphe d'adjacence construit sur le plan précédent. Les ouvertures sont en pointillées. Nous pouvons ainsi remarquer que le graphe de voisinage n'est en général pas un sous-graphe du graphe d'adjacence, puisque deux cellules peuvent être voisines sans qu'il existe une ouverture entre les deux (cas des cellules C7 avec C10, et C2 avec C5).

3.3.2 Le graphe de visibilité

Dès lors que le graphe d'adjacence est établi, nous pouvons calculer l'ensemble des cellules visibles depuis une cellule donnée en parcourant ce graphe depuis la cellule initiale. Chaque arc emprunté correspond à une ouverture entre deux cellules. Pour que la cellule atteinte soit visible de notre cellule de départ, il suffit donc de tester leur visibilité à travers la séquence d'ouvertures traversée. Lorsque ces calculs de visibilité sont terminés, nous obtenons alors le graphe de visibilité où:

– un sommet du graphe représente une cellule;

– un arc relie deux sommets si les cellules correspondantes sont visibles entre elles.

En premier lieu, nous pouvons déjà retenir, sans aucun test, les cellules adjacentes (reliées par un arc) comme étant visibles. En effet, d'après la définition du graphe d'adjacence, ces cellules ont une ouverture commune avec notre cellule de départ. Elles sont donc visibles à travers ces ouvertures. Cette propriété est illustrée par la figure 3.14. Sur cet

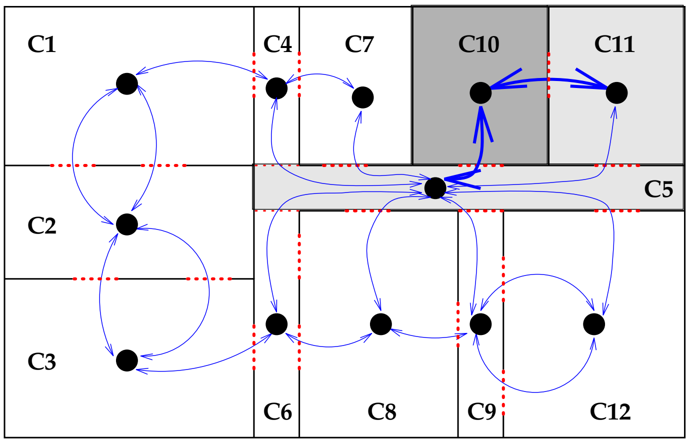

Fig. 3.14 - *Visibilité des cellules adjacentes*

exemple, nous voulons calculer les cellules visibles de la cellule C10. Les cellules adjacentes C5 et C11 peuvent donc déjà être marquée comme visibles. Deux séquences d'ouvertures sont dans le même temps initialisées. Nous nous intéressons par la suite à la séquence correspondant au départ par l'ouverture entre les cellules C10 et C11.

Nous devons maintenant progresser dans notre graphe, en essayant toutes les ouvertures de la cellule atteinte (exceptée l'ouverture d'arrivée). Sur notre exemple, une seule ouverture permet d'avancer: c'est celle menant à la cellule C5. Pour tester la visibilité de cette nouvelle cellule, il suffit de tester la visibilité de l'ouverture qui nous a permis d'y entrer. Si l'ouverture est visible, alors la cellule le sera également.

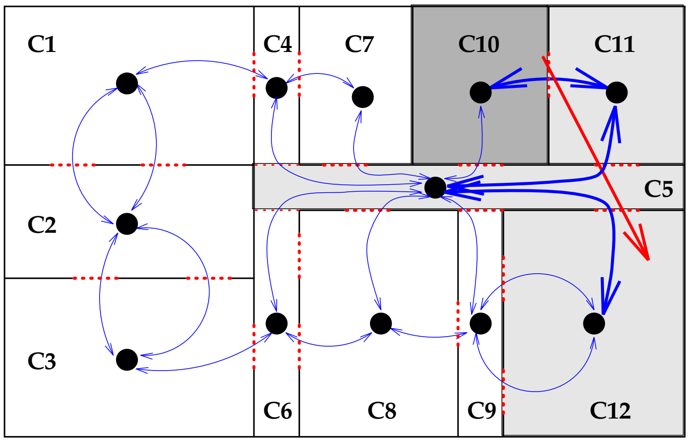

Fig. 3.15 - *Visibilité inter-cellules par lancer de rayon*

Pour résoudre ce problème de visibilité, nous avons choisi une méthode basée sur un lancer de rayons, similaire à celle d'Airey (voir chapitre 2). Cette méthode consiste à lancer aléatoirement des rayons entre la première et la dernière ouverture traversées. Si un seul rayon intersecte toutes les ouvertures, alors la visibilité des deux cellules est établie. C'est le cas par exemple des cellules C5 et C12 sur la figure 3.15. Le rayon de visibilité (représenté par une flèche sur la figure) intersecte bien toutes les ouvertures traversées pour atteindre ces cellules.

Le choix du nombre maximum de rayons à lancer est paramétrable par l'utilisateur. Cependant, plus il est important, plus les calculs de visibilité seront longs. En effet, deux cellules sont considérées comme invisibles (par la séquence d'ouvertures traversée) une fois que le nombre maximum de rayons de visibilité est atteint. Mais si l'on diminue ce nombre, le risque de ne pas retenir toutes les cellules visibles augmente. Ce problème est bien entendu lié à l'échantillonnage des ouvertures déjà évoqué section 2.3.

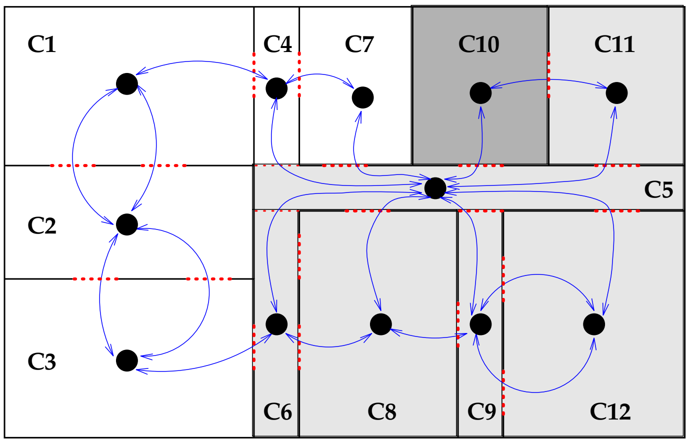

FIG. 3.16 - *Cellules visibles d'une cellule initiale*

Lorsque toutes les séquences d'ouvertures possibles ont été essayées depuis la cellule de départ, l'algorithme s'arrête. L'ensemble des cellules visibles de la cellule C10 est représenté figure 3.16.

3.4 Structures essentielles

Dans cette section, nous présentons les structures de données importantes de nos algorithmes: les informations contenues dans un noeud de l'arbre BSP seront exposées section 3.4.1. Les feuilles de l'arbre contiennent seulement une référence aux cellules obtenues.

Nous détaillerons la structure des cellules section 3.4.2. Nous verrons notamment comment les graphes de voisinage, d'adjacence et de visibilité sont représentés dans ces cellules. Enfin, nous examinerons section 3.4.3 les différents fichiers générés.

3.4.1 L'arbre BSP

La structure de données d'un noeud BSP est déclarée comme suit:

```
typedef struct NOEUD {
  union {
        ENGLOBANT *englobant;
        CELLULE *cellule;
  } info;
  PLAN *plan;
  LFACETTE *lf_plan;
  structNOEUD *pere;
  structNOEUD *coteneg;
  structNOEUD *cotepos;
} BSP_AXIAL;
```

Le champ *info* contient les informations relatives à un noeud (terminal ou non). Dans le cas d'une feuille, ce champ associe les informations relatives à une cellule. Dans le cas d'un noeud (non terminal), il associe les informations relatives à une boite englobante.

Le noeud racine de l'arbre correspond à la boite englobante axiale de toute la scène. Comme les plans de découpage sont tous axiaux, les régions situées de part et d'autre du plan de découpage sont encore des boites axiales. Finalement, chaque noeud de l'arbre BSP représente une sous-boite axiale de la scène. (*englobant*) est donc un pointeur sur la sous-boite représentée par le noeud. Seuls deux sommets suffisent pour décrire complètement cette boite.

Exceptées les feuilles de l'arbre BSP, tous les noeuds contiennent un plan découpant la boite représentée. La variable *plan* pointe donc sur le plan de découpage retenu. Un plan est décrit par ses paramètres cartésiens.

Le plan de découpage calculé supportent des facettes contenues dans la boite *englobant*. Ces facettes constitueront les facettes murales des cellules finales obtenues. Afin de les insérer dans ces cellules, nous devons les stocker temporairement sur les noeuds de l'arbre BSP. *lf_plan* est donc la liste des facettes présentes sur le plan.

Un lien vers le noeud père (pointeur *pere*) est nécessaire pour calculer les relations de voisinages entre cellules (voir section 3.2.5.1). Seule la racine de l'arbre n'a pas de lien père.

Finalement, deux liens (pointeurs *coteneg* et *cotepos*) sont nécessaires pour pouvoir générer de façon récursive l'arbre BSP. Ces liens sont créés dès qu'un nouveau plan de découpage est introduit. *coteneg* mène à la boite (représentée par un nouveau noeud) située du côté négatif du plan, alors que *cotepos* mène au côté positif.

Si les deux pointeurs *coteneg* et *cotepos* sont nuls, alors le noeud est une feuille de l'arbre. Cette feuille représente la boite englobante d'une cellule. Dans ce cas, le pointeur *englobant* est remplaçé par le pointeur *cellule*. Ce dernier pointe sur une structure CELLULE (cette structure est détaillée section 3.4.2).

3.4.2 Les cellules

La structure de données d'une cellule est déclarée comme suit:

```
typedef struct CELLULE {
    UNSIGNED num;
    ENGLOBANT englobant;
    TAB_MUR_CELLULE lf_eng;
    PTR_LFACETTE lf_inclus;
    TAB_OUVERTURE_CELLULE lo_eng;
    TAB_CELLULE_VOISINE lc_voisine;
    STRUCT LCELLULE *lcellule_visible;
    STRUCT NOEUD *pere;
    INT marque;
} CELLULE;
```

Une cellule est identifiée par un numéro (variable *num*). Ce numéro correspond à son ordre de création. Il servira d'identificateur dans les fichiers décrivant le graphe de visibilité.

La variable *englobant* pointe sur la boite englobante de la cellule.

Les facettes murales de la cellule sont référencées dans la variable *lf_eng*. Cette variable est un tableau de 6 cases représentant les 6 côtés de la cellule. Pour chaque côté de la cellule, nous avons la liste des facettes présentes sur ce côté.

lf_inclus est la liste des facettes strictement incluses dans la cellule. Ces facettes sont les facettes axiales non contenues dans des plans de découpages, ainsi que les facettes non traitées par l'agorithme (facettes non axiales ou non rectangulaires).

Les ouvertures de la cellule sont également référencées dans un tableau de 6 cases (*lo_eng*) pour les 6 côtés de la cellule. Chaque case de ce tableau contient la liste des ouvertures calculées sur un côté de la cellule.

Les cellules voisines de la cellule sont, elles aussi, référencées dans un tableau de 6 cases (*lc_voisine*) pour les 6 côtés de la cellule. Chaque case du tableau contient donc la liste des cellules voisines sur un côté de la cellule.

lcellule_visible est la liste des cellules visibles.

Pour rechercher les cellules voisines, il est nécessaire de pouvoir remonter dans l'arbre BSP à partir d'une cellule (voir section 3.2.5.1). *pere* pointe donc directement sur le noeud père du noeud correspondant à la cellule (feuille de l'arbre). Ce lien est généré à la création de la cellule.

Enfin, *marque* est un champ d'information supplémentaire pour la cellule: il servira notamment à marquer les cellules vides. Une cellule vide est une cellule ne contenant pas de facettes (ni murales, ni incluses). Le programme ne calcule pas les cellules visibles depuis une cellule vide. De plus, les cellules vides ne sont pas insérées dans l'ensemble des cellules visibles depuis une cellule. Le problème des cellules vides est largement détaillé au chapitre 4.

3.4.3 Les fichiers générés

Plusieurs fichiers peuvent être générés par notre programme. Leur nombre dépend des options choisies dans le fichier de configuration (voir Annexe A). Les noms de ces fichiers sont également précisés dans le fichier de configuration. Les différents formats utilisés pour ces fichiers sont détaillés à l'annexe B.

3.4.3.1 Les fichiers cellules

Lorsque l'étape de structuration de la scène est terminée, nous obtenons une liste de cellules. Chaque cellule représente un ensemble de facettes contenues dans une boite englobante. Ces facettes sont les facettes murales de la cellule ainsi que les facettes incluses. Rappelons que l'objectif de cette structuration est d'obtenir un graphe de visibilité. La structure de ce graphe de visibilité est sauvée dans un fichier (voir section suivante) et les informations relatives à chaque noeud (les facettes de chaque cellule) sont stockés dans des fichiers cellules. Il est ainsi possible de ne charger que l'ensemble des fichiers correspondant aux cellules visibles d'une cellule donnée.

Un fichier cellule contient la description (géométrique) de toutes ses facettes. Les cellules vides (c'est à dire ne contenant pas de facettes) ne sont pas écrites dans un fichier.

Le programme écrit donc autant de fichiers cellules que de cellules non vides obtenues. La syntaxe utilisée est décrite section B.1.

3.4.3.2 Le graphe de visibilité

Le fichier contenant le graphe de visibilité indique pour chaque cellule, l'ensemble des cellules non vides qui lui sont visibles. La syntaxe utilisée est décrite section B.2.

3.4.3.3 Les fichiers de contrôle

Deux types de fichiers de contrôle peuvent être générés. Le premier type (fichier de contrôle global) sert à visualiser le résultat de l'étape de structuration de la scène en cellules. Le second type (fichier de contrôle visibilité) sert à visualiser l'ensemble des cellules visibles depuis une cellule donnée.

Pour visualiser les différentes cellules obtenues, il suffit d'affecter la même couleur aux facettes appartenant à la même cellule. En prenant soin de ne pas avoir deux cellules voisines de la même couleur, la structuration obtenue par l'algorithme ressort ainsi clairement. Le fichier de contrôle global contient donc toutes les facettes issues de la structuration, colorée selon cette méthode.

Un fichier de contrôle visibilité cellule affecte également des couleurs aux facettes de la scène:

1. Les facettes de la cellule controlée sont écrites en rouge.
2. Les facettes des cellules visibles sont écrites en dégradé de vert (du vert clair au vert foncé selon la distance de cette cellule à la cellule contrôlée).
3. Les facettes des autres cellules sont écrites dans des couleurs différentes en respectant cependant une même couleur par cellule.

Il est ainsi facile de voir les cellules visibles d'une cellule donnée. Il y aura donc autant de fichiers de contrôle visibilité que de fichiers cellules.

La syntaxe utilisée pour les fichiers de contrôle est décrite section B.2.

3.4.3.4 Les fichiers statistiques

Les fichiers statistiques sont générés quand le programme est lancé en mode statistique. Dans ce mode, plusieurs paramètres concernant la structuration sont évalués en fonction de la valeur du degré d'opacité minimum (critère principal) et du critère de sélection des plans de découpage (critères secondaires) retenus. Ces paramètres sont détaillés section 5.1. Un fichier statistique correspond à la variation d'un paramètre en fonction du critère principal et pour un critère secondaire donné.

La syntaxe utilisée pour les fichiers statistiques est décrite section B.4.

3.4.3.5 Les fichiers MatLab

Pour visualiser sous forme graphique la variation des paramètres mesurés, nous avons choisi d'utiliser MatLab. Ce logiciel permet de créer de nouvelles commandes sous la forme de fichiers. Un fichier de commande MatLab est donc générer pour représenter graphiquement les variations d'un paramètre. Nous aurons donc autant de fichiers MatLab que de paramètres mesurés.

Ces fichiers sont chargés d'afficher sous forme de courbe les valeurs contenues dans les fichiers statistiques écrits par le programme.

Le contenu des fichiers de commande MatLab est décrit section B.5.

Chapitre 4

Difficultés rencontrées

Au chapitre 2, nous avons vu que la complexité du calcul des visibilités est liée au nombre de facettes à traiter. Pour visualiser une scène de taille importante, nous structurons celle-ci en cellules et calculons les relations de visibilité entre cellules. Ainsi, nous réduisons l'ensemble des facettes potentiellement visibles (ou EFPV) d'un observateur: seules les facettes appartenant aux cellules visibles de la cellule contenant l'observateur peuvent être affichées à l'écran. Dans ce chapitre, nous nous intéressons à l'influence de la structuration sur ces EFPV.

Certaines cellules nécessitent d'être à nouveau découpées pour àméliorer les calculs de visiblité. Ce nouveau découpage utilise des critères différents des critères présentés section 3.2.2. Ainsi, les cellules trop grandes limitent parfois sérieusement l'intérêt de la structuration de la scène pour les calculs de visibilité. De même, l'aspect d'une cellule détermine de façon importante l'efficacité des algorithmes de visibilité. Ces deux problèmes sont étudiés section 4.1 et 4.2.

La visibilité des cellules entre elles n'est possible qu'à travers une séquence d'ouvertures. Cependant, les ouvertures détectées sur la frontière des cellules conduisent parfois à rendre visibles des cellules voisines qui ne le sont pas en réalité. En étudiant l'origine de ces *fausses* ouvertures, nous proposons une méthode, section 4.3, pour ne plus en avoir.

4.1 Des cellules trop grandes

L'intérêt de la structuration en cellules est de limiter au maximum les ensembles de facettes potentiellement visibles d'un observateur. Pourtant, si une cellule est trop volumineuse, seule une petite partie de celle-ci risque d'être visible d'une autre cellule.

Considérons la structuration obtenue sur le plan de la figure 4.1. Les cellules (nommées C1 à C4) sont en pointillés. La cellule C1 comporte, de plus, deux objets composés chacun

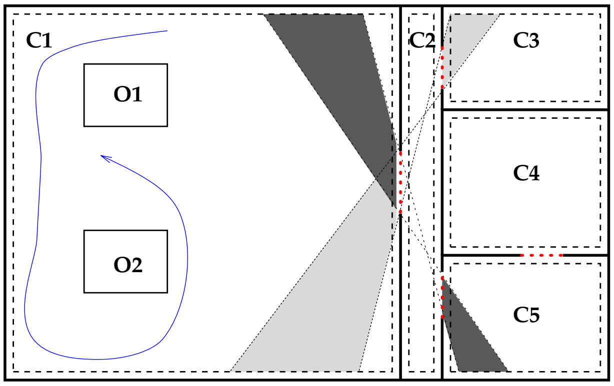

FIG. 4.1 - *Cellule trop grande*

de plusieurs milliers de facettes (seuls les boites englobantes de ces objets sont représentées). Si nous nous intéressons à la visibilité de la cellules C1 depuis la cellule C3, nous pouvons constater que seule la zone gris clair de C1 est potentiellement visible depuis C3. Les objets O1 et O2 sont en particulier complètement invisibles depuis C3. Malgré cela, toutes les facettes appartenant à C1 devront être considérées pour visualiser la scène depuis C3. Le même problème surgit pour la visibilité de C1 depuis C5. Seule la zone gris foncé de C1 est potentiellement visible de C5, et les objets O1 et O2 ne sont toujours pas visibles.

Dans les deux cas, la surestimation des ensembles de facettes potentiellement visibles est beaucoup trop importante. Les calculs de visibilité nécessaires pour visualiser la scène depuis les cellules C3 ou C5 ne bénéficient pratiquement pas de la structuration effectuée. Ce dernier point reste également vrai si l'on considère maintenant la visibilité de la cellule C1. Depuis C1, les cellules C2, C3 et C5 peuvent être visibles, c'est à dire pratiquement toute la scène. Pourtant, un observateur se déplaçant le long de la trajectoire indiquée sur la figure ne verra jamais les cellules C3 et C5, et les facettes de ces cellules continueront à être traitées par l'algorithme de visibilité final.

Parce qu'on ne précalcule pas des informations de visibilité cellules à régions (au sens parties d'une cellule), le problème des cellules trop grandes se pose de façon cruciale. En effet, la visibilité entre deux cellules n'est possible qu'à travers une séquence d'ouvertures. Or ce sont ces ouvertures qui limitent cette visibilité à certaines parties des cellules. Plus la séquence comporte d'ouvertures et plus les parties visibles sont réduites.

Malgré cela, seule l'information indiquant la visibilité potentielle ou non entre deux cellules est calculée. De ce fait, les facettes considérées comme potentiellement visibles entre elles sont toutes les facettes contenues dans les deux cellules. Si une cellule représente une grande partie de la scène (comme la cellule C1 sur la figure), la complexité du calcul des visibilités n'est plus vraiment diminuée, et la structuration obtenue ne présente plus d'intérêt.

Au chapitre 2, nous avons vu que seul Teller détermine une solution analytique au problème de la visibilité à travers une séquence d'ouvertures. Mais cette approche reste mathématiquement complexe et demande plusieurs heures de précalcul sur des grosses scènes. Nous préférons une approche plus simple: découper les cellules volumineuses en cellules plus petites. De cette façon, les ensembles de facettes potentiellement visibles à considérer seront certainement plus restreints.

De plus, cette méthode présente l'avantage de diminuer le nombre d'ouvertures potentielles dans ces cellules. Nous parlons d'ouvertures potentielles parce que celles-ci ne peuvent être calculées qu'après avoir terminé le découpage de la scène en cellules (voir section 3.2.5). En diminuant le nombre des ouvertures par cellule, nous diminuons d'autant le nombre de séquences d'ouvertures potentielles partant d'une cellule. Les calculs de visibilité entre cellules sont donc accélérés.

Avant de découper ces cellules trop volumineuses, essayons de connaître les raisons de leur trop grande taille. Pour cela, il faut revenir à l'étape sélectionnant un plan de découpage pendant la décomposition BSP de la scène. Nous ne retenons que les plans dont le degré d'opacité est supérieur à un seuil fixé à priori. Il existe donc deux raisons pour ne plus retenir un plan:

- soit il n'y a plus de facette dans la cellule courante;
- soit les plans supportant ces facettes ont un degré d'opacité trop faible.

Dans le cas où il reste encore des facettes, nous pouvons toujours diminuer le seuil du degré d'opacité minimum jusqu'à ce qu'un plan de découpage soit candidat. Cela revient à choisir le plan ayant le plus haut degré d'opacité. Cependant, le risque augmente de découper selon des facettes ne représentant plus un élément structurant majeur de la scène (tels que les murs, plafonds, sols) mais appartenant à des objets plus petits (composant le mobilier par exemple). Il n'est surement pas intéressant de découper un amphithéatre par un plan supportant la face d'une boite d'allumettes ! En effet, ce plan entraine un sur-découpage inutile de la scène alors que la boite d'allumettes n'est pas occlusive. De plus, la structuration devient complexe si cette boite d'allumettes se déplace.

Pour limiter ces problèmes, il est possible d'indiquer un seuil minimum pour le degré d'opacité. Si aucun plan dont le degré d'opacité est supérieur à ce seuil, nous pouvons considérer la cellule comme n'ayant plus de facettes.

Si la cellule n'a plus de facettes (ou est considérée comme telle), le plan de découpage pourrait être choisi n'importe où dans la cellule. Cependant, la forme de la cellule doit être examinée pour ne pas obtenir un aspect de la cellule trop petit (le problème de l'aspect d'une cellule est présenté section 4.2). Nous avons donc choisi dans ce cas de découper la cellule en son milieu et selon un plan perpendiculaire à son côté le plus long.

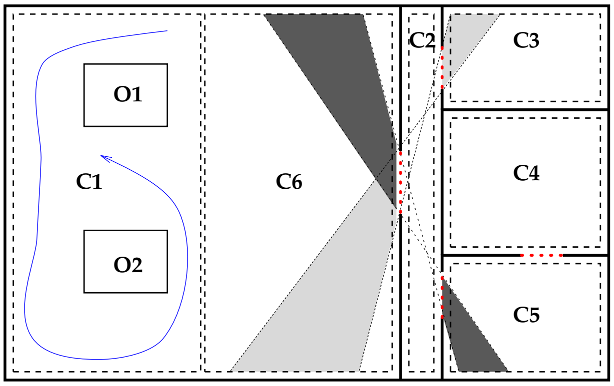

FIG. 4.2 - *Découpage d'une cellule trop grande*

La figure 4.2 montre ainsi le découpage de la cellule initiale C1 en deux cellules plus petites C1 et C6. Ainsi, la cellule C1 et les objets O1 et O2 sont totalement invisibles des cellules C3 et C5. Les ensembles de facettes potentiellement visibles depuis C3 ou C5 sont ainsi très largement réduits. L'algorithme de visibilité, traitant moins de facettes, est nettement accéléré. De même, notre observateur se déplaçant uniquement dans la cellule C6, les cellules C3 et C5 ne seront pas pris en compte pendant la visualisation de la scène.

4.2 L'aspect des cellules

L'aspect d'une cellule mesure sa forme dans une direction axiale donnée (voir section 3.1.3). Si une cellule est trop allongée, une faible partie de celle-ci est visible depuis une autre cellule (non voisine).

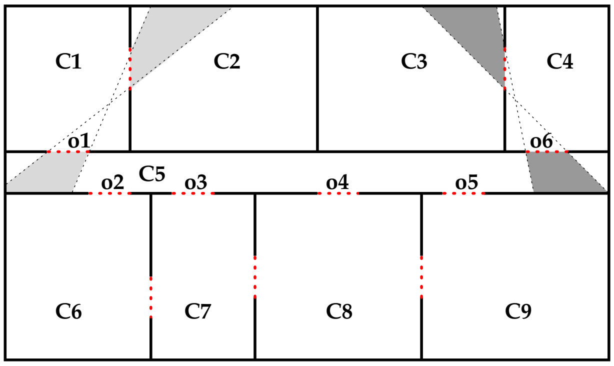

FIG. 4.3 - *Cellule trop allongée*

Ce problème est illustré par la figure 4.3. Sur cet exemple, l'étape de structuration a donné 9 cellules (nommées C1 à C9). Nous pouvons constater que la cellule C5 est cependant très allongée (ce qui peut-être vérifié par un aspect très petit). Notons que malgré sa longueur, son volume est sensiblement le même que celui des autres cellules. Elle n'a donc pas pu être découpée sur le critère du volume. Intéressons-nous à la visibilité de cette cellule depuis les cellules C2 et C3. Dans les deux cas, les parties de C5 visibles sont très petites (elles sont représentées par des zones grisées). Nous retrouvons ici le même problème que dans le cas des cellules trop grandes: trop de facettes sont considérées comme visibles alors qu'elles ne le sont pas.

Mais il existe un autre problème inhérent à la forme allongée d'une cellule: le nombre de cellules qui lui sont visibles peut suffir parfois à rendre la structuration totalement inutile. En effet, revenons à la cellule C5 de notre figure, et regardons quelles cellules lui sont visibles. Au travers des ouvertures o1 à o6, les cellules C1, C4, C6, C7, C8 et C9 sont visibles. De plus, nous avons vu que les cellules C2 et C3 lui sont également visibles. Finalement, toute la scène est visible depuis la cellule C5. Une visualisation de la scène depuis cette cellule n'exploitera donc en rien la structuration obtenue. Cet exemple n'est pas un cas isolé. Très souvent, les différentes pièces d'un étage sont toutes disposées de part et d'autre d'un long couloir. Il existe alors autant d'ouvertures dans ce couloir que de pièces attenantes.

Même si nous pouvons ignorer le premier problème en raison du volume finalement restreint de ces cellules, le second problème doit être résolu sous peine d'obtenir une structuration complètement inexploitable. Notre solution est de découper cette cellule perpendiculairement à son coté le plus long. L'aspect d'une cellule nous indique quel est ce côté. Une position de découpe optimale sur ce côté reste cependant difficile à trouver. Ainsi, une analyse de la répartition des ouvertures sur cette cellule permettrait certainement d'obtenir des cellules ayant un visibilité réduite. Malheureusement, les ouvertures ne sont calculées qu'après avoir terminé le découpage de la scène en cellules, à l'aide du graphe de voisinage (voir section 3.2.5). Finalement, nous avons choisi de découper la cellule en son milieu.

4.3 Les fausses ouvertures

Les cellules obtenues par notre algorithme de découpage BSP rassemblent les facettes appartenant à une même boite englobante. Un des objectifs de ce découpage est de retrouver une structuration qui suivent les éléments occlusifs majeurs de la scène. Par exemple, l'idéal est qu'une pièce non convexe soit représentée par une cellule. Dans ce cas, les facettes murales correspondent réellement aux murs de la pièce, et les ouvertures délimitent bien les portes et fenêtres de cette pièce.

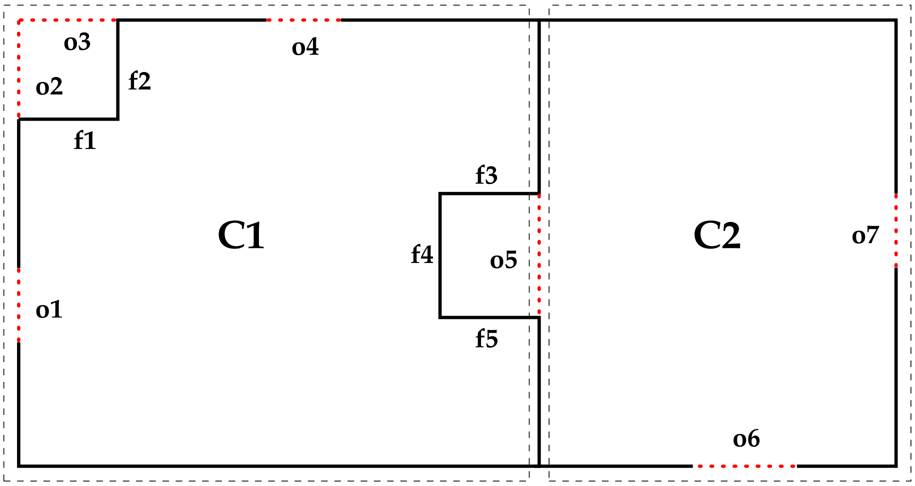

FIG. 4.4 - *Exemple d'une pièce non convexe*

Cependant, il existe des cas où la cellule obtenue, bien qu'englobant une pièce de la scène, ne reflète en rien sa forme. Il suffit que la pièce soit non convexe. La figure 4.4 montre le découpage en cellule obtenu sur deux pièces voisines. Les frontière des cellules sont enpointillé. La cellule C1 englobe une pièce non convexe. En effet, celle-ci présente un coin rentrant (en haut à gauche, facettes f1, f2, f3) et un renfoncement (au milieu à droite, facettes f3, f4, f5). La cellule C2 englobe une pièce convexe.

Le problème se situe à la détection des ouvertures. Nous avons vu (section 3.2.5) que les ouvertures correspondent aux zones de la frontière de la cellule dépourvues de facettes. Sur la figure, les ouvertures calculées sont les ouvertures o1 à o7. Mais seules les ouvertures o1, o4, o6 et o7 sont valables. Les autres ouvertures sont qualifiées de **fausses ouvertures** et doivent être éliminées. En effet, depuis ces ouvertures la pièce compris dans la cellule C1 reste complètement invisible. Elles ne peuvent donc pas être utilisées pour calculer l'ensemble des cellules visibles de la cellule. Cela conduirait, par exemple, à considérer que la cellule C2 est liée à la cellule C1 dans le graphe d'adjacence, et donc augmenterait inutilement l'ensemble des cellules visibles par C1.

Pour résoudre ce problème, deux approches sont possibles:

- soit détecter puis éliminer les fausses ouvertures après leur construction;
- soit continuer à découper les cellules pour ne jamais en avoir.

La première solution implique de tester la visibilité des ouvertures avec les facettes composant la cellule (murales et incluses). Un lancer de rayon permettrait ainsi de détecter les fausses ouvertures.Si aucun rayon, lancé depuis l'ouverture et vers la cellule, ne rencontre en premier une facette (de la cellule) du côté de sa normale, alors l'ouverture est éliminée. Cependant, le risque de trouver une ouverture fausse alors qu'elle est correcte existe. En effet, le nombre de rayons lancés est forcément fini et l'ouverture est elle-même échantillonnée. De plus, ces nouveaux calculs de visibilité rendrait l'algorithme plus complexe et surtout moins rapide.

Nous préférons alors la seconde solution qui, même si elle génère un surplus important de nouvelles cellules et ouvertures, donne un résultat exact sur leurs ouvertures. Pour ne plus avoir de fausses ouvertures, il suffit de ne plus avoir de coins rentrants ou renfoncements dans les cellules. Les coins rentrants et les renfoncements sont provoqués par les facettes axiales incluses dans la cellule (différentes des facettes murales) et intersectant sa frontière. Ainsi, sur la figure 4.4, les facettes f1, f2, f3 et f5 sont des facettes incluses intersectant la frontière de la cellule (en pointillé). La cellule est alors découpée selon des plans supportant ces facettes. Le plan ayant le plus fort degré d'opacité est sélectionné en priorité.

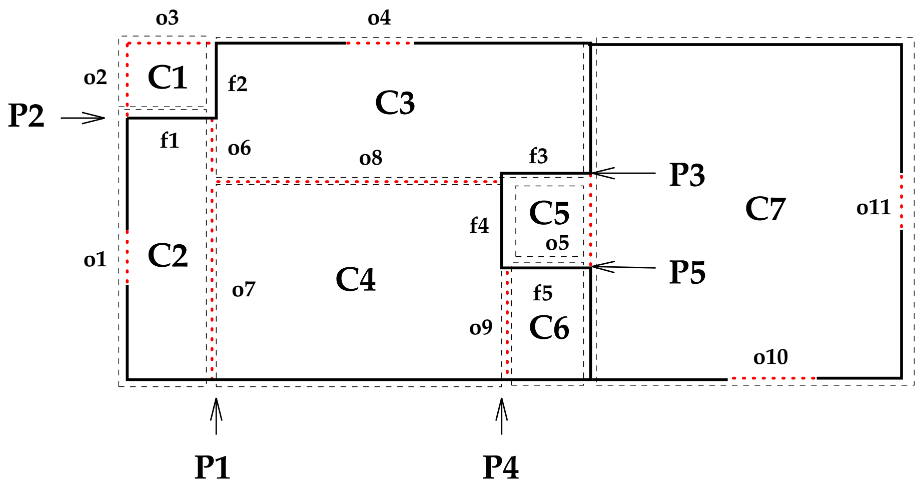

FIG. 4.5 - *Découpage d'une pièce non convexe*

La figure 4.5 nous montre le résultat du découpage de la pièce non convexe de la figure 4.4. Les plans de découpage choisit sont nommés en fonction de leur ordre de sélection. Nous pouvons constater que le nombre de cellules est passé à 6, et que 4 nouvelles ouvertures sont apparues. De plus, les cellules C1 et C5 sont vides. Néanmoins, il n'y a plus de fausses ouvertures.

Chapitre 5

Résultats

Le but de nos travaux a été de réduire la complexité du calcul des visibilités en structurant les scènes. L'objectif est que cette structuration suive au mieux la topologie des scènes traitées. Plusieurs paramètres permettent d'évaluer l'influence des critères de découpage retenus dans la structuration d'une scène et la construction du graphe de visibilité. Ils sont introduits en section 5.1. Pour illustrer les travaux réalisés, nous avons pris trois exemples de scène architecturale de complexité croissante. Leur description détaillée ainsi que les différents résultats obtenus sont présentés en section 5.2.

5.1 Les paramètres mesurés

Les différents paramètres mesurés portent sur l'étape de structuration d'une part, et sur la construction du graphe de visibilité d'autre part. Les premiers mesurent l'influence des critères de découpage sur la structuration de la scène. Ils sont présentés section 5.1.1. Les seconds mesurent l'efficacité de l'algorithme construisant le graphe de visibilité. Ils sont introduits section 5.1.2. Ces mesures permettent, en particulier, d'apprécier l'importance d'une bonne structuration pour construire un graphe de visibilité efficace.

5.1.1 Statistiques de structuration

Les statistiques de structuration mesurent plusieurs paramètres concernant l'arbre BSP, les cellules et les ouvertures obtenus. Le critère de découpage principal étant le degré d'opacité minimum, nous avons mesuré les variations de plusieurs paramètres en fonction de ce critère. Ces mesures sont facilitées par un mode d'exécution particulier du programme (mode $STAT$, voir annexe A), qui permet d'obtenir plusieurs valeurs d'un même paramètre en variant le degré d'opacité minimum. De plus, nous voulions comparer les trois critères secondaires entre eux (choix du plan médian, du plan d'opacité maximum, du plan découpant le moins de facettes). Pour chaque paramètre mesuré, nous obtenons finalement trois séries de valeurs (une série par critère secondaire) représentant sa variation en fonction du degré d'opacité minimum.

Concernant l'arbre BSP, les paramètres suivants sont mesurés (en fonction du degré d'opacité minimum):

- le nombre de plans de découpage introduits, ou encore, le nombre de noeuds (différents d'une feuille) présents dans l'arbre;
- la profondeur maximum de l'arbre;
- la profondeur moyenne de l'arbre;
- le pourcentage de facettes sur les plans de découpage;
- le pourcentage d'augmentation du nombre de facettes;
- le temps d'exécution pris pour établir l'arbre BSP.

Les profondeurs maximum et moyennes de l'arbre permettent de contrôler son équilibrage.

Le pourcentage de facettes sur les plans de découpage nous donne une indication sur le nombre de facettes qui ont été trouvées comme éléments structurants majeurs d'une scène. Si la scène ne contient pas d'objets (si elle n'est composées que de murs, plafonds et sols), ce nombre doit donc être le plus élevé possible. En effet, en découpant au maximum selon les murs d'une scène, la topologie de la scène (une cellule par pièce par exemple) sera mieux conservée.

Le pourcentage d'augmentation du nombre de facettes témoigne du nombre de facettes coupées pendant la décomposition BSP. Plus ce nombre est important, plus la taille mémoire nécessaire pour traiter la scène augmente.

Plusieurs paramètres concernant les cellules sont également mesurés:

- le nombre de cellules obtenues (ce nombre correspond aussi au nombre de feuilles dans l'arbre BSP);
- le pourcentage de cellules vides;
- le pourcentage de cellules avec facettes incluses;
- le pourcentage de cellules avec facettes murales;
- le temps d'exécution pour calculer les facettes murales.

Rappelons qu'une cellule avec facettes murales est une cellule contenant des facettes sur sa frontière. Nous les appelerons également cellules murales.

Les cellules avec facettes incluses sont les cellules contenant des facettes autres que murales. Nous les appelerons également cellules peuplées.

Une cellule vide ne contient ni facettes incluses, ni facettes murales. Leur pourcentage nous renseigne sur leur importance dans la scène. Ainsi, nous verrons que plus leur nombre est important, plus les temps de calcul du graphe de visibilité augmentent.

Enfin, pour les ouvertures, nous mesurons:

- le nombre d'ouvertures obtenues;
- le pourcentage d'ouvertures entre cellules vides;
- le pourcentage d'ouvertures de cellules vides à cellules pleines;
- le pourcentage d'ouvertures entre cellules pleines;
- le temps d'exécution pour calculer les ouvertures.

Le nombre d'ouvertures obtenues nous indique directement le nombre d'arcs dans le graphe d'adjacence. Plus ce nombre est important, plus les parcours de recherche des cellules visibles sont nombreux, et dans ce cas plus les calculs contruisant le graphe de visibilité sont longs.

La répartition des ouvertures selon le type des cellules qu'elles relient nous permet, d'une part, d'apprécier la qualité du découpage et, d'autre part, d'évaluer le nombre d'arcs du graphe d'adjacence effectivement parcourus.

Ainsi, en découpant uniquement selon les plans des éléments structurants majeurs d'une scène (murs, sols et plafonds), un bon découpage donnerait des cellules pleines pour les pièces et les embrasures des portes et fenêtres, et des cellules vides pour les régions de la scène situées dans l'épaisseur des murs par exemple. De plus, et à moins qu'aucune pièce ne soit découpée en plusieurs cellules, il ne devrait pas y avoir de possibilités pour passer d'une cellule vide à une cellule pleine (ou inversement) en traversant une ouverture.

Imaginons, en effet, une personne désirant quitter une pièce: en général, elle choisiera de passer par une porte ou une fenêtre plutôt que de jouer au "passe-muraille". Ainsi, elle quittera une première cellule murale (la pièce de départ), passera par une seconde cellule murale plus petite (l'embrasure de la porte ou de la fenêtre), pour finalement arriver dans une troisième cellule murale (pièce voisine ou couloir par exemple). Les ouvertures traversées sont uniquement des ouvertures entre cellules pleines.

Le pourcentage d'ouvertures de cellules vides à cellules pleines permet donc d'apprécier la qualité du découpage. Notons enfin qu'une scène mal modélisée peut également générée de telles ouvertures. Il suffit, par exemple, que des "trous" apparaissent au niveau des jointures entre les facettes représentant les murs.

Finalement, le pourcentage du nombre d'ouvertures entre cellules pleines nous donnera une indication sur le nombre d'arcs du graphe d'adjacence pouvant être traversés en partant d'une cellule pleine. Bien entendu, cette dernière valeur est d'autant plus significative que le pourcentage d'ouvertures entre cellules vides et cellules pleines est faible.

5.1.2 Calculs de visibilité

Le graphe de visibilité représente les relations de visibilité existantes entre les cellules issues de l'étape de structuration. Notre algorithme construisant ce graphe de visibilité lance des rayons entre les ouvertures des cellules. De la même manière que pour les statistiques de structuration, nous mesurons différents paramètres en fonction du critère d'opacité minimum et du critère secondaire. Les paramètres mesurés sont:

- le nombre de rayons lancés;
- le nombre de tests de visibilité inter-ouvertures;
- le pourcentage de tests réussis;
- le nombre de liens de visibilité générés;
- le temps d'exécution pour calculer le graphe de visibilité.

Le pourcentage de tests réussis indique le nombre de tests positifs de visibilité entre deux ouvertures par rapport au nombre de tests total.

Le nombre de liens générés correspond au nombre d'arcs dans le graphe de visibilité.

5.2 Scènes traitées

Tous les tableaux statistiques introduits dans les sections suivantes ont la même structure: la première colonne décrit les paramètres mesurés, puis les 9 colonnes suivantes contiennent leurs valeurs respectives.

Les colonnes des valeurs sont divisées en 3 groupes. Chaque groupe correspond à une valeur précise du degré d'opacité minimum (critère secondaire). Cette valeur est indiquée sur la première ligne du tableau.

De plus, chaque groupe est lui-même divisé en trois colonnes nommées C1, C2 et C3 (sur la seconde ligne du tableau). Ces colonnes indiquent quel critère secondaire (choix du plan de découpage) est retenu:

- la colonne C1 correspond au choix du plan médian;
- la colonne C2 correspond au choix du plan d'opacité maximum;
- la colonne C3 correspond au choix du plan découpant le moins de facettes.

Pour mesurer les différents paramètres ci-dessus, trois scènes de complexité croissante sont présentées.

La première, *le loft*, est un petit appartement de deux pièces meublées. Une description détaillée et les résultats obtenus sur cette scène sont présentés section 5.2.1.

La seconde, *l'étage*, représente un étage entier, composé principalement de bureaux. Cette scène est introduite section 5.2.2.

La troisième, *l'immeuble*, correspond à un immeuble de cinq étages (dont l'étage de la seconde scène). Cette dernière scène est un modèle de l'U.C. BERKELEY COMPUTER SCIENCE BUILDING[1]. Les résultats de nos travaux sur cette scène sont présentés section 5.2.3.

5.2.1 Scène 1: le loft

Une description détaillée de cette scène est présentée en section 5.2.1.1. Les résultats statistiques sur la structuration de cette scène sont présentés en section 5.2.1.2. Ceux concernant la construction du graphe de visibilité sont introduits en section 5.2.1.3.

5.2.1.1 Description de la scène

Le *loft* est une scène composée de deux pièces meublées. Elle comporte en tout 175 facettes dont 111 sont rectangulaires et axiales. Le nombre total de positions axiales est de 80. La figure 5.1 montre une vue en fil de fer de cette scène.

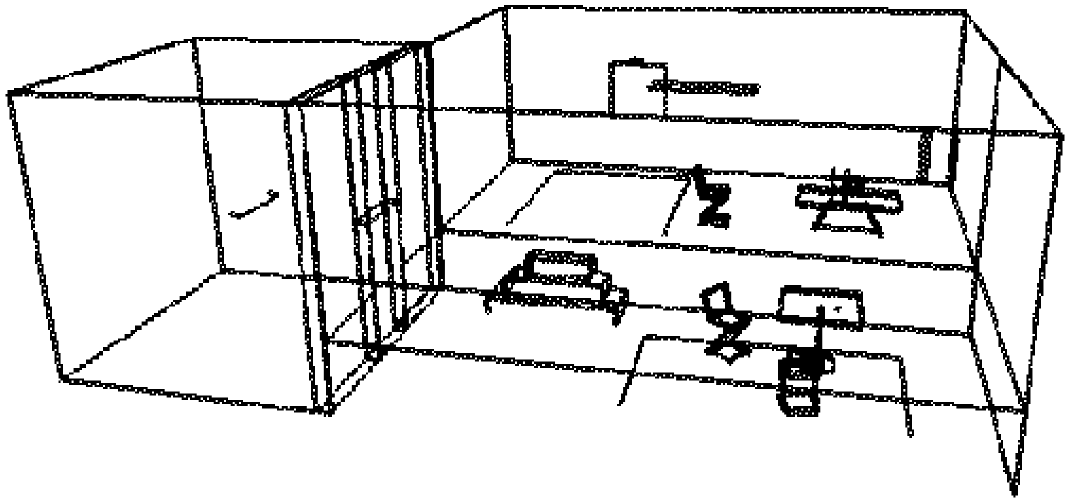

FIG. 5.1 - *Scène 1: le loft*

1. distribué par Seth Teller sur sa page personnelle WWW: **http://theory.lcs.mit.edu/šeth/**

5.2.1.2 Analyse de la structuration

Nous présentons ici plusieurs mesures sur la structuration de la scène du *loft*. Ces mesures sont réparties dans trois tableaux: le tableau 5.1 regroupe les paramètres sur le découpage BSP, le tableau 5.2 regroupe ceux sur les cellules et le tableau 5.3 regroupe ceux sur les ouvertures.

	Opacité = 0			**Opacité = 0.3**			**Opacité = 0.7**		
	C1	C2	C3	C1	C2	C3	C1	C2	C3
Nb de noeuds	499	227	185	45	31	29	33	31	31
Nb de feuilles	250	114	93	23	16	15	17	16	16
Prof. maximum	13	26	28	9	12	12	11	12	11
Prof. moyenne	9	15	16	6	7	7	7	7	7
% faces supp.	196.6	148	121.1	6.8	6.8	6.8	6.8	6.8	6.8
% faces sur plans	73.2	69.6	64.6	18.7	18.7	18.7	18.7	18.7	18.7
Temps calcul BSP	< 1 s.								

TAB. 5.1 - *Statistiques du découpage BSP du loft*

D'emblée, nous pouvons constater que les valeurs du premier groupe du tableau 5.1 sont toutes très supérieures à celles des deux autres groupes. Ainsi, par exemple, le nombre de noeuds dans l'arbre BSP passe de 499 pour le critère C1 du premier groupe, à respectivement 45 et 33 pour le même critère dans les deux autres groupes.

Cette différence importante est la conséquence du choix pris pour la valeur minimum du degré d'opacité d'un plan de découpage. En effet, le nombre de noeuds dans l'arbre BSP correspond au nombre de plans de découpage introduits pendant la décomposition de la scène. En prenant un degré d'opacité minimum de 0, tous les plans supportant des facettes axiales de la scène peuvent alors être candidats pour découper la scène. Ainsi, par exemple, tous les plans supportant les facettes axiales composant les objets (chaise, table, boite d'allumettes) de la scène, seront choisis pour découper la scène.

Ceci est d'ailleurs confirmé en regardant le pourcentage des facettes de la scène présentes sur les plans de découpage: cette valeur passe ainsi de plus de 60 % dans le premier groupe (et même 73.2 % pour le critère C1, soit près des trois-quarts des facettes de la scène), à seulement 18.7 % pour les deux autres groupes.

En augmentant le nombre de plans de découpages (et donc de noeuds dans l'arbre), le nombre de cellules (les feuilles de l'arbre) augmente également. Ainsi, 250 cellules sont générées pour le critère C1 du premier groupe, alors qu'elles ne sont plus respectivement que 23 et 17 pour le même critère dans les deux autres groupes. Nous sommes donc loin, ici, de suivre la topologie de la scène qui n'est composée que de 2 pièces!

De plus, en augmentant le nombre de cellules, nous augmentons aussi le nombre d'ouvertures calculées sur les frontières de ces cellules. Ainsi, plusieurs centaines d'ouvertures sont calculées pour le premier groupe du tableau 5.3, contre une trentaine seulement pour les deux autres groupes.

	Opacité = 0			**Opacité = 0.3**			**Opacité = 0.7**		
	C1	C2	C3	C1	C2	C3	C1	C2	C3
Nb cellules	250	114	93	23	16	15	17	16	16
% cell. vides	43.2	12.3	12.9	69.6	56.2	53.3	58.8	56.2	56.2
% cell. murales	55.6	85.9	87.1	30.4	43.8	46.7	41.2	43.8	43.8
% cell. peuplées	10.4	18.4	20.4	13.0	18.8	20	17.6	18.8	18.8
Temps calcul murs	< 1 s.								

TAB. 5.2 - *Statistiques des cellules du loft*

L'augmentation du nombre de plans de découpage provoque naturellement plus de coupures sur les facettes d'une scène. Ainsi, le pourcentage d'augmentation du nombre de facettes après la construction de l'arbre dépasse les 100% (et avoisine même les 200% avec le critère C1) dans le premier groupe, alors qu'il n'est plus que de 6.8% dans les deux autres groupes. Rappelons que l'objectif principal de la structuration est de diminuer la complexité du calcul des visibilités dans les scènes. Nous avons vu que cette complexité est en particulier due au nombre de facettes d'une scène. Il s'agit donc de limiter au maximum son augmentation pendant la phase de structuration.

Ces premiers résultats confirment donc l'importance du critère d'opacité minimum dans le choix des plans de découpage candidats.

Mais ce critère n'est pas le seul à influer sur la structuration. En effet, en comparant les valeurs obtenues pour chaque critère secondaire (C1, C2 et C3), nous pouvons également remarquer d'importantes variations. Ces variations sont d'ailleurs d'autant plus importantes que le degré d'opacité minimum est faible. Ainsi, dans le premier groupe (où le degré d'opacité minimum est nul), le nombre de feuilles (soit cellules) obtenues est divisé par 2.5 entre les critères C1 et C3, et le pourcentage d'augmentation du nombre de facettes est réduit de 196.6 à 121.1.

Dans le même temps, les pourcentages de cellules vides (resp. murales et peuplées) et d'ouvertures entre cellules vides (resp. pleines) diminuent (resp. augmentent). Le problème des cellules vides (resp. ouvertures entre cellules vides) est qu'elles sont également représentées par des sommets (resp. des arcs) dans le graphe d'adjacence. De ce fait, le nombre de sommets (et dans le même temps, d'arcs) parcourus dans le graphe d'adjacence pour calculer les cellules visibles d'une cellule donnée augmente. Finalement, les temps de calculs pour construire le graphe de visibilité augmentent également.

Pour expliquer la variation des paramètres en fonction des critères C1, C2 et C3, il faut revenir à leur définition. Ces critères sont utilisés pour sélectionner un plan parmi les plans de découpages candidats (i.e. retenus par le critère d'opacité minimum), afin de subdiviser une cellule.

Ainsi, le critère C1 retient le plan le plus médian des plans candidats pour découper la cellule courante. L'objectif recherché est d'équilibrer au maximum l'arbre BSP final. Pour apprécier cet équilibrage, nous pouvons comparer les profondeurs moyennes et maximales obtenues. Plus l'écart est important, plus l'arbre est déséquilibré. En comparant les profondeurs maximum et moyennes obtenues pour les trois critères, nous pouvons constater que leur écart est beaucoup moins important avec le critère C1, qu'avec les deux autres critères. De ce fait, l'arbre BSP est mieux équilibré avec C1 qu'avec C2 ou C3.

Mais, le critère C1 ne tient pas compte des conséquences du choix du plan de découpage sur les cellules découpées. Ainsi par exemple, le degré d'opacité obtenu entre les deux nouvelles cellules (chacune d'un côté du plan de découpage) n'est pas analysé.

Le critère C2 sélectionne le plan dont le degré d'opacité est maximum. De ce fait, les plans supportant les facettes de petite taille (composant les objets, par exemple) ne sont pas retenus en priorité. Le découpage s'effectue donc d'abord selon les principaux éléments structurants de la scène (murs, plafonds, etc.).

Ce choix permet donc de ne pas couper dès le début de grandes facettes par des plans supportants de petites facettes. Le nombre de facettes coupées est donc diminué (par exemple, pour le premier groupe, 148 % d'augmentation du nombre de facettes avec C2 contre 196.6 % avec C1). Comme il y a moins de facettes dans la scène, il y aura moins de plans de découpage candidats (227 avec C2 contre 499 avec C1), et donc moins de cellules obtenues (114 avec C2 contre 250 avec C1).

De plus, comme les plans de découpage ont un degré d'opacité plus élevé, nous obtenons plus de cellules murales, c'est à dire contenant des facettes sur leur frontière (et donc, dans le même temps, moins de cellules vides). Parallèlement, le nombre d'ouvertures entre cellules vides (resp. pleines) diminue (resp. augmente) d'autant.

Les résultats sont encore améliorés (de façon moins importante cependant) avec le critère C3. Contrairement au critère C2, le critère C3 choisit directement le plan découpant le moins de facettes. De ce fait, le pourcentage d'augmentation du nombre de facettes est le plus bas parmi les trois critères (121.1 pour le premier groupe). Les nombre de plans de découpages et de cellules sont donc encore réduits.

	Opacité = 0			Opacité = 0.3			Opacité = 0.7		
	C1	C2	C3	C1	C2	C3	C1	C2	C3
Nb ouvertures	754	383	293	43	28	26	29	26	26
% vides-vides	35.9	5.7	7.8	86	78.6	76.9	79.3	76.9	76.9
% pleines-pleines	62.9	91.9	90.1	14	21.4	23.1	20.7	23.1	23.1
% pleines-vides	1.2	2.3	2.0	0	0	0	0	0	0
Temps calcul ouv.	< 1 s.								

TAB. 5.3 - *Statistiques des ouvertures du loft*

Quand le degré d'opacité minimum augmente, les écarts entre les valeurs obtenues par les critères C1, C2, et C3 sont réduits. Ceci s'explique par le fait que beaucoup moins de plans sont retenus par le critère principal (opacité minimum). En effet, seuls les plans dont l'opacité est supérieur au seuil indiqué par ce critère peuvent être candidats, ce qui élimine automatiquement les plans supportant des facettes de petites dimensions. Dans ce cas, seuls les éléments structurants de la scène sont retenus. Ce dernier point explique les valeurs identiques (18.7%) concernant le pourcentage de faces sur les plans de découpage dans les second et troisième groupes.

Notons, enfin, que les temps de calcul pour construire l'arbre BSP, calculer les murs des cellules et calculer les ouvertures sont tous inférieurs à 1 seconde. Le peu de facettes contenues dans la scène explique cette rapidité.

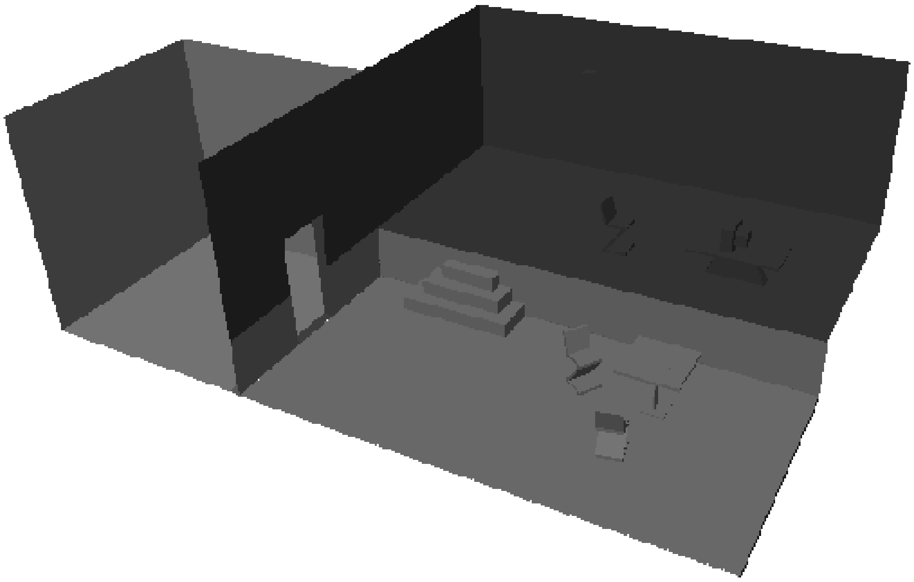

Fig. 5.2 - *Structuration obtenue sur le loft*

La figure 5.2 nous montre la structuration obtenue sur *le loft* en prenant la valeur 0.3 pour le degré d'opacité minimum et en choisissant les plans découpant le moins de facettes.

Nous pouvons distinguer 4 cellules murales:

- une première suivant parfaitement la pièce de gauche;
- une seconde suivant l'embrasure de la porte;
- une troisième délimitant le niveau inférieur de la grande pièce;
- une quatrième délimitant le niveau supérieur de la grande pièce.

Remarquons, enfin, que les plans des facettes composant l'escalier et le mobiler n'ont pas été retenus par l'algorithme.

5.2.1.3 Analyse du graphe de visibilité

Le tableau 5.4 regroupe les paramètres concernant le calcul du graphe de visibilité des cellules.

Encore une fois, les valeurs du premier groupe sont extrêmement différentes des deux autres groupes. Le nombre d'ouvertures calculées pendant l'étape de structuration est ici déterminant pour les calculs de visibilité. En effet, rappelons que la relation de visibilité entre deux cellules est établie à travers une séquence d'ouvertures. De ce fait, plus le

	Opacité = 0			Opacité = 0.3			Opacité = 0.7		
	C1	C2	C3	C1	C2	C3	C1	C2	C3
Nb rayons lancés	13175888	6179751	3534926	58	58	55	64	61	59
Nb tests ouv.-ouv.	1475023	697720	403838	18	18	18	18	18	18
% tests réussis	15.4	16	18.1	77.8	77.8	77.8	77.8	77.8	77.8
Nb liens générés	12452	5934	4012	40	40	40	40	40	40
Temps visibilité	371 s.	177 s.	97 s.	< 1 s.					

TAB. 5.4 - *Statistiques de visibilité du loft*

nombre d'ouvertures dans la scène est important, plus les séquences d'ouvertures risquent d'être longues. Dans ce cas, le nombre de tests de visibilité inter-ouvertures augmente également. Ce dernier point est confirmé par les valeurs du tableau: le nombre de tests inter-ouvertures diminue avec le nombre d'ouvertures.

Pour comprendre les faibles pourcentages de réussite pour déterminer la visibilité de deux ouvertures, il faut se rappeler l'algorithme utilisé. Partant d'un noeud dans le graphe d'adjacence (une cellule dans la scène), nous parcourons en profondeur toutes les suites d'arcs possibles (séquences d'ouvertures). Arrivé sur un noeud (cellule), soit nous trouvons que la cellule correspondante est visible de la cellule de départ, soit non. Dans le premier cas, la recherche continue en parcourant les arcs quittant le noeud courant (les ouvertures de la cellule). Dans le second cas, la recherche s'interrompt et reprend au noeud précédent (cellule précédente) avec un arc (ouverture) non encore parcouru. Comme le nombre de cellules réellement visibles est très inférieur au nombre de cellules traversées, le pourcentage de tests réussis reste très faible.

Pour tester la visibilité des deux cellules, nous testons la visibilité des deux ouvertures extrêmes de la séquence des ouvertures traversées entre les deux cellules. Notre algorithme lance un nombre fini de rayons entre deux ouvertures pour déterminer leur visibilité. Si un rayon traverse toutes les ouvertures de la séquence, alors les deux ouvertures sont visibles. Si aucun rayon intersectant toutes les ouvertures n'est trouvé, alors elles sont invisibles. Ici, dix rayons maximum ont été lancés entre deux ouvertures. Cela signifie que pour déterminer que deux ouvertures sont invisibles, il faut nécessairement lancer ces dix rayons. Le nombre de rayons lancés est d'autant plus grand que le nombre de tests manqués est important. Ceci explique les valeurs importantes pour le nombre de rayons lancés et les temps de calculs correspondants dans le tableau.

5.2.2 Scène 2: l'étage

Une description détaillée de *l'étage* est présentée section 5.2.2.1. Les résultats statistiques sur la structuration de cette scène sont présentés section 5.2.2.2. Ceux concernant la construction du graphe de visibilité sont introduits section 5.2.2.3. Enfin, certains cas particuliers sont mis en évidence à la section 5.2.2.4.

5.2.2.1 Description de la scène

L'étage est une scène composée d'une cinquantaine de pièces, de couloirs et corridors, dénués de tout mobilier. Elle comporte en tout 2003 facettes dont 1956 sont rectangulaires et axiales. Le nombre total de positions axiale est de 497. La figure 5.3 montre une vue en fil de fer de cette scène.

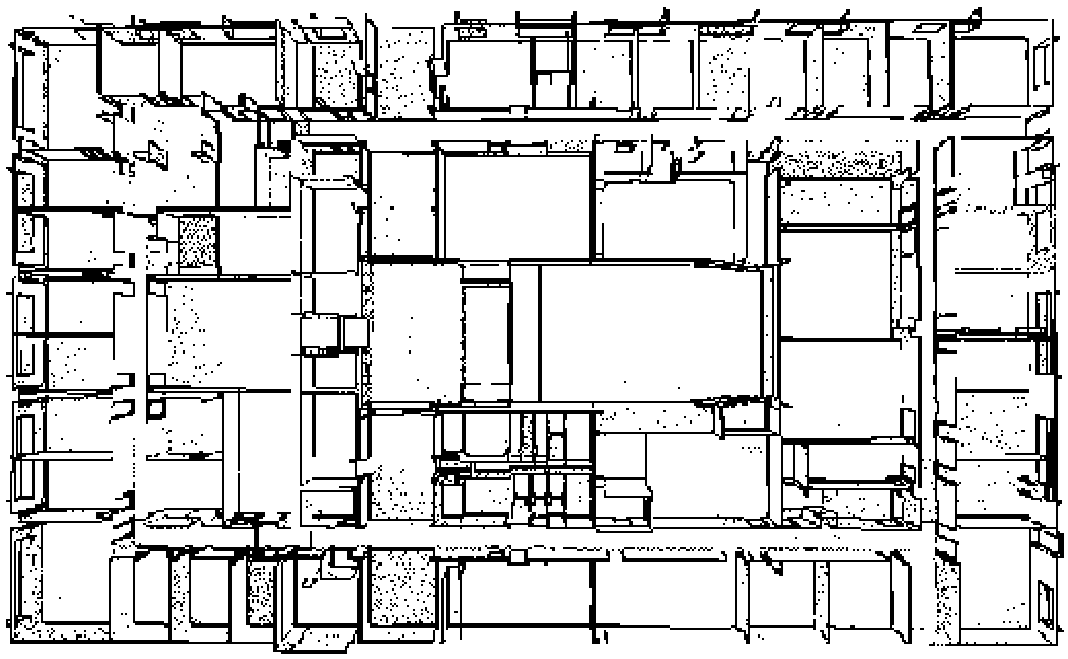

FIG. 5.3 - *Scène 2: l'étage*

5.2.2.2 Analyse de la subdivision

Nous présentons ici plusieurs mesures sur la structuration de la scène de *l'étage*. Ces mesures sont réparties dans trois tableaux: le tableau 5.5 regroupe les paramètres sur le découpage BSP, le tableau 5.6 regroupe ceux sur les cellules et le tableau 5.7 regroupe ceux sur les ouvertures.

	Opacité = 0			**Opacité = 0.4**			**Opacité = 0.6**		
	C1	C2	C3	C1	C2	C3	C1	C2	C3
Nb de noeuds	5793	2367	2071	2425	2035	1419	11	11	11
Nb de feuilles	2897	1184	1036	1213	1018	710	6	6	6
Prof. maximum	17	39	46	28	39	34	6	6	6
Prof. moyenne	13	24	26	16	24	22	4	4	4
% faces supp.	116.8	42.9	25.8	46.3	40.4	22.1	1.2	1.2	1.2
% faces sur plans	97.0	96.4	96.9	90.7	89.7	78.7	19.0	19.0	19.0
Temps calcul BSP	2 s.	2 s.	3 s.	1 s.	1 s.	2 s.	< 1 s.		

TAB. 5.5 - *Statistiques du découpage BSP de l'étage*

Tout comme pour la scène du *loft*, nous pouvons remarquer l'extrême variabilité des données entre les trois groupes des tableaux.

Les valeurs du troisième groupe des tableaux (où le degré d'opacité minimum est fixé à 0.6) sont identiques pour les trois critères C1, C2 et C3. Ainsi, 6 cellules sont générées: 4 sont murales et 2 sont peuplées. Les 4 cellules murales correspondent aux 4 cellules longeant les 4 côtés de la scène (les facades extérieures servant de facettes murales). Les 2 cellules peuplées se partagent l'intérieur de l'étage et près de 81 % des facettes restantes (19 % des facettes étant murales).

Finalement, 30 liens sont générés entre ces cellules, c'est à dire que toutes les cellules sont visibles entre elles.

Ce découpage est donc peu intéressant puisqu'il ne réduit pas les ensembles de visibilité des cellules. Intéressons-nous au deux premiers groupes des tableaux.

Les résultats du second groupe apparaissent nettement meilleurs que ceux du premier. Ainsi, pour un même critère, le nombre de cellules et le nombre des ouvertures calculés sont diminués. De même, le pourcentage d'augmentation du nombre de facettes est réduit (il n'est plus que de 22.1% avec C3 dans le second groupe).

En revanche, les pourcentages extrêmement élevés des facettes présentes sur les plans de découpage dans le premier groupe (près de 97% avec C1, C2 ou C3) sont légèrement diminués dans le second groupe (moins de 90%). Ceci s'explique bien entendu par un nombre de plans de découpage candidats inférieurs en raison du degré d'opacité minimum exigé. Certaines facettes restent donc peuplées les cellules.

	Opacité = 0			**Opacité = 0.4**			**Opacité = 0.6**		
	C1	C2	C3	C1	C2	C3	C1	C2	C3
Nb cellules	2897	1184	1036	1213	1018	710	6	6	6
% cell. vides	58.4	55.3	58.4	52.6	52.7	54.2	0	0	0
% cell. murales	41.4	44.4	41.4	47.2	46.9	45.2	83.3	83.3	83.3
% cell. peuplées	1.5	2.11	1.35	2.7	3.14	2.7	33.3	33.3	33.3
Temps calcul murs	< 1 s.								

TAB. 5.6 - *Statistiques des cellules de l'étage*

Que ce soit dans le premier ou le second groupe, la répartition des cellules entre cellules vides et cellules murales varie peu: environ 55% des cellules sont vides, contre 45% de cellules murales.

Comme les murs de la scène ont tous une épaisseur, deux pièces reliées par une porte donneront 3 cellules:

- une cellule pour la première pièce;
- une cellule correspondant à l'embrasure de la porte;
- une cellule pour la seconde pièce.

Deux ouvertures seront également créées:

- une ouverture reliant la première cellule à la seconde;
- une ouverture reliant la seconde cellule à la troisième.

De la même manière, les embrasures des fenêtres constitueront des cellules, et une ouverture sera créée entre cette cellule et la cellule représentant la pièce attenante.

Les cellules vides se trouvent principalement dans les murs de la scène. En effet, comme les murs ont une épaisseur, en découpant la scène selon ses murs, nous générons automatiquement des cellules entre ces murs. Or les facettes de ces murs sont orientées vers les pièces attenantes, et constituent leurs facettes murales. Les cellules situées dans les murs sont ainsi dépourvues de facettes.

Finalement, le nombre de cellules vides dépend donc étroitement du nombre de cellules murales et ceci explique les répartitions identiques entre cellules vides et murales trouvées dans le tableau.

En revanche, le pourcentage des ouvertures entre cellules vides diminue pratiquement de moitié au profit de celui des ouvertures reliant deux cellules pleines ou une cellule vide et une cellule pleine. Ceci s'explique par le fait qu'en augmentant l'opacité minimum, on sélectionne plus de plans correspondants à des éléments structurants importants de la scène (tels que les murs) plutôt que des plans supportant des facettes de petites dimensions (rebord de fenêtre par exemple). De ce fait, les cellules obtenues suivent mieux les pièces de la scène. Dans ce cas, les ouvertures relieront plus souvent des pièces entre elles (au sens pièce - embrasure porte - pièce) que des zones vides de la scène.

	Opacité = 0			**Opacité = 0.4**			**Opacité = 0.6**		
	C1	C2	C3	C1	C2	C3	C1	C2	C3
Nb ouvertures	8002	3413	3123	3448	2824	2270	1345	1345	1345
% vides-vides	62.1	67.4	69.8	47.9	36.7	36.2	0	0	0
% pleines-pleines	29.2	21.2	17.9	29.2	29.8	32.8	100	100	100
% pleines-vides	8.7	11.4	12.3	22.9	33.5	31.0	0	0	0
Temps calcul ouv.	27 s.	5 s.	5 s.	5 s.	3 s.	3 s.	12 s.	12 s.	12 s.

TAB. 5.7 - *Statistiques des ouvertures de l'étage*

La figure 5.4 montre une structuration obtenue sur l'étage avec un degré d'opacité minimum de 0.4 et le critère C3 (choix du plan découpant le moins de facettes). Les facettes d'une même cellule sont de la même couleur. On peut constater que dans l'ensemble les cellules obtenues suivent les pièces de l'étage.

La figure 5.5 montre une telle pièce avec 5 portes. Nous pouvons ainsi constater que les embrasures de ces portes constituent également des cellules.

La figure 5.6 nous montre la vue d'une des façades de l'étage. Nous pouvons remarquer que les fenêtres de cette façade sont également des cellules, distinctes des cellules représentant les pièces attenantes.

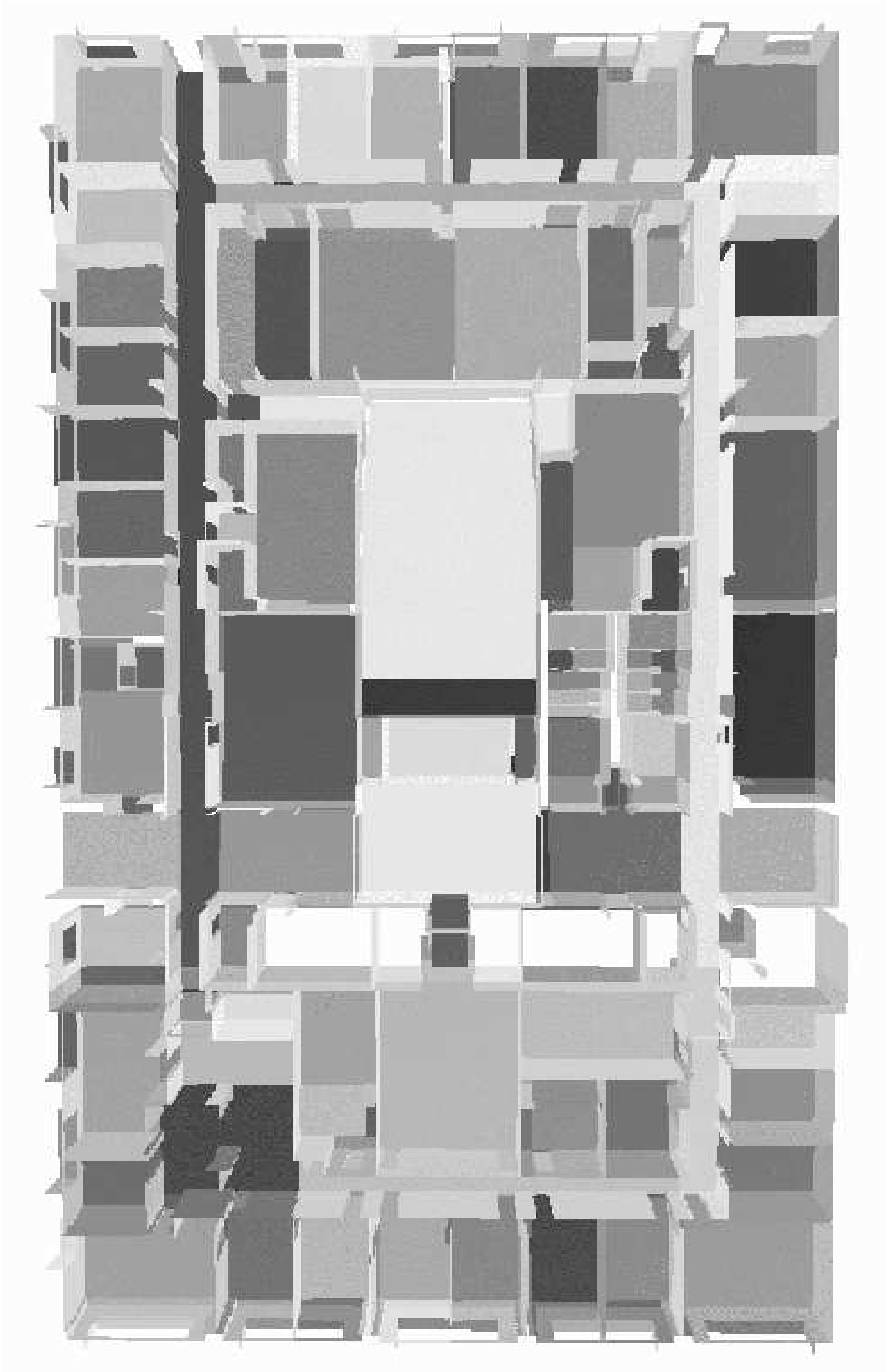

FIG. 5.4 - *Structuration obtenue sur l'étage*

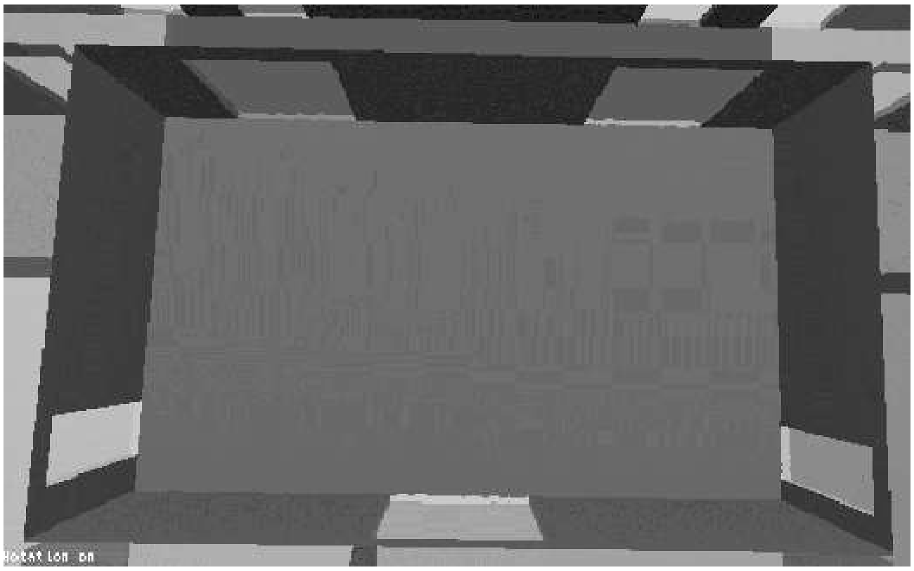

FIG. 5.5 - *Vue d'une piece et de ses ouvertures*

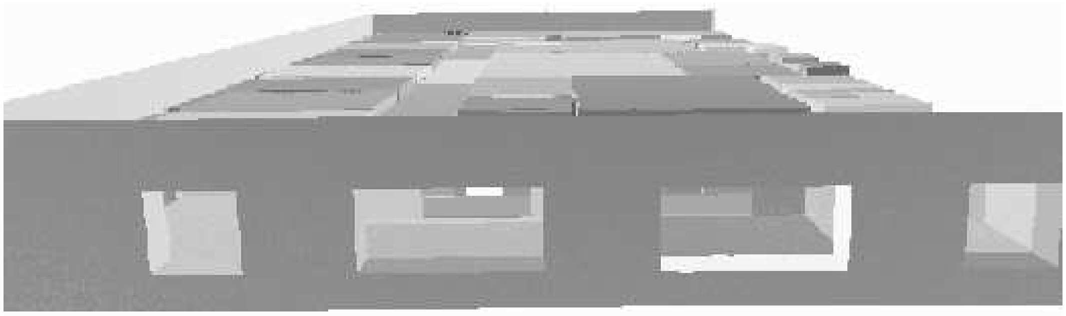

FIG. 5.6 - *Vue d'une façade de l'étage*

5.2.2.3 Analyse du graphe de visibilité

Le tableau 5.8 présente les résultats concernant le calcul du graphe de visibilité sur *l'étage*.

	Opacité = 0			**Opacité = 0.4**			**Opacité = 0.6**		
	C1	C2	C3	C1	C2	C3	C1	C2	C3
Nb rayons ($\times 10^6$)	41.7	12.5	13.8	39.8	13.5	28.4	4.3	3.9	2.6
Nb tests ($\times 10^5$)	48.5	13.7	15.3	42.5	14.8	30.3	5.6	5.2	3.8
% réussis	19.8	11.9	13.3	8.8	11.2	8.6	26.4	27.5	36.7
Nb liens	94564	21478	14684	33032	18772	12118	30	30	30
Temps (en s.)	4738	1235	1468	3237	1061	2363	309	312	191

TAB. 5.8 - *Statistiques de visibilité de l'étage*

Les différentes variations des paramètres sont identiques à celles notées pour le *loft*: moins il y a d'ouvertures, moins de liens de visibilité sont générés. En revanche, les temps de calculs ne sont plus négligeables. Ainsi, la construction du graphe de visibilité de *l'étage* peut prendre jusqu'à 80 minutes (avec le degré d'opacité minimum nul et le critère C1).

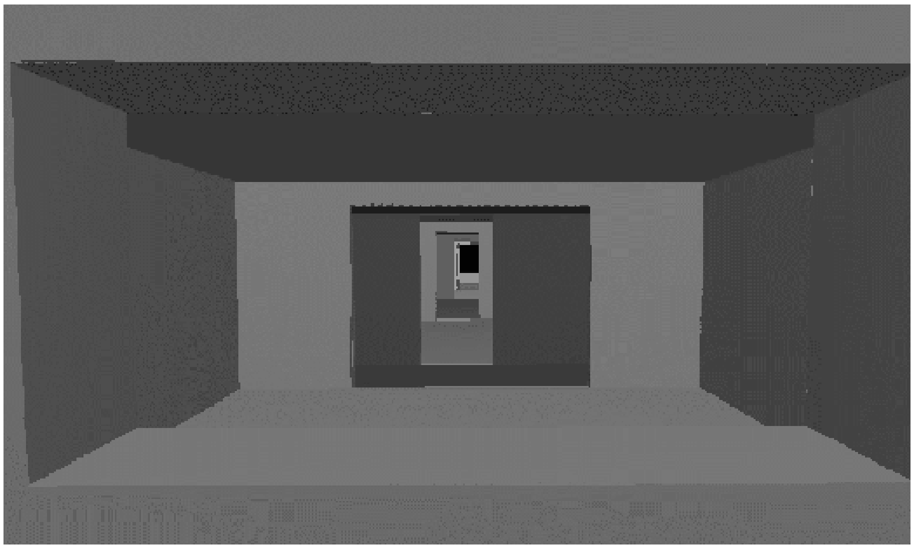

FIG. 5.7 - *Vue à travers une séquence d'ouvertures. Cette vue est prise depuis une des fenêtres de la façade extérieure.*

5.2.2.4 Cas particuliers

Nous commentons ici quelques cas particuliers de structuration de *l'étage*.

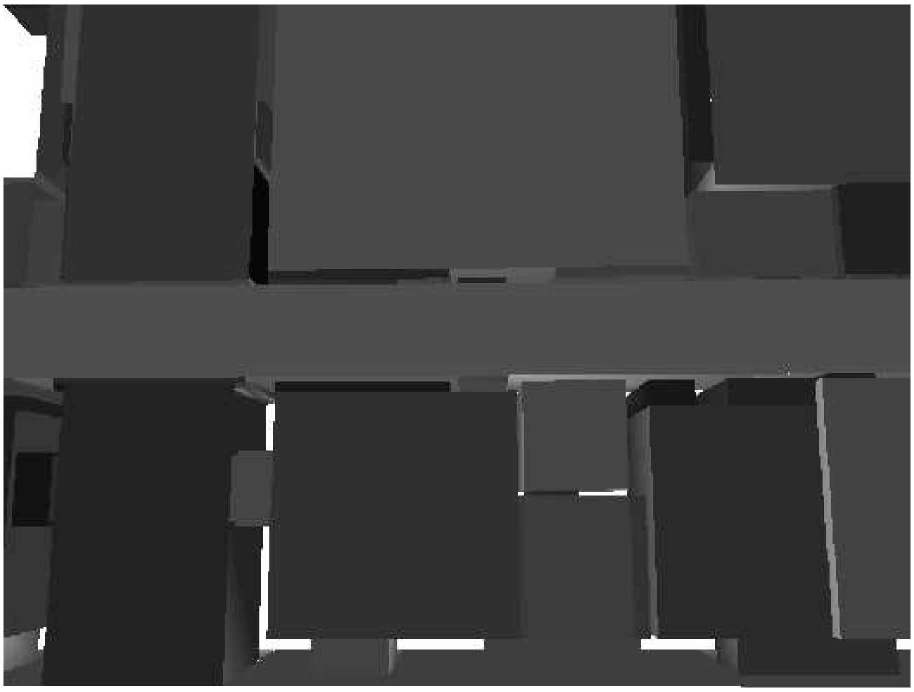

FIG. 5.8 - *Vue rapprochée du toit de l'étage*

La figure 5.8 nous présente une vue rapprochée du toit de l'étage. En premier lieu, nous pouvons remarquer que les cellules suivent parfaitement les pièces et les embrasures des portes et fenêtres de l'étage.

De plus, l'épaisseur des murs est ici bien apparente. Les cellules vides issues de la structuration se trouvent principalement dans les interstices situés entre les pièces.

Enfin, le couloir est représenté par une seule cellule car l'option de découpage basée sur l'aspect des cellules a été désactivée.

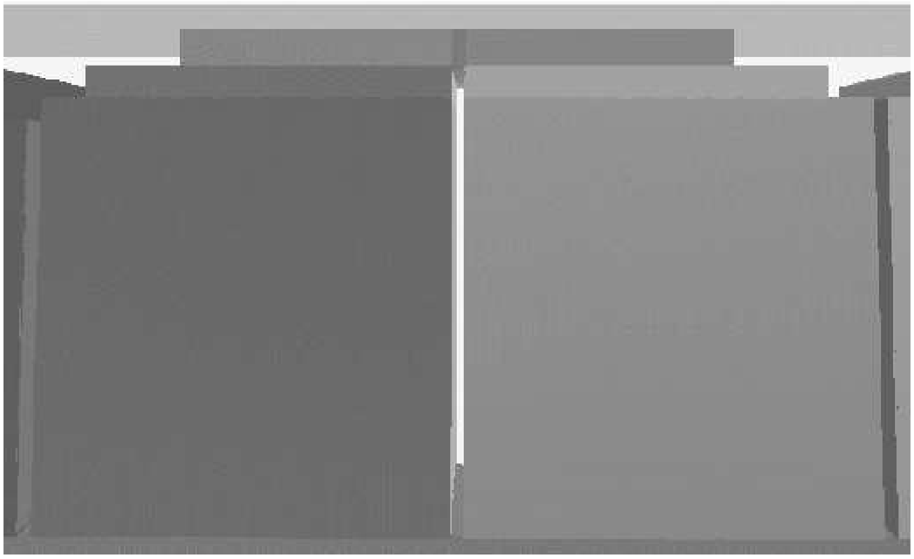

FIG. 5.9 - *Vue de deux pièces non convexes*

Nous pouvons de plus remarquer que certaines pièces non convexes ont été découpées (selon la méthode présentée section 4.3) pour éviter de générer de fausses ouvertures. La figure 5.9 illustre ainsi le découpage obtenue sur deux pièces non convexes. En partant du haut de cette figure, nous pouvons constater plusieurs cellules:

- la façade est visible et rassemble les facettes murales d'une première cellule;
- nous trouvons ensuite deux cellules correspondants aux embrasures des fenêtres des pièces;
- puis, deux cellules assez allongées témoignent du découpage des pièces non convexes;
- enfin, les deux cellules les plus importantes représentent pratiquement les deux pièces initiales.

Notons la présence d'une petite cellule entre les deux cellules principales: c'est la cellule correspondant à l'embrasure d'une porte reliant les deux pièces. Enfin, nous pouvons clairement distinguer l'épaisseur des murs, dans lesquels se situent des cellules vides.

5.2.3 Scène 3: l'immeuble

Une description détaillée de *l'immeuble* est présentée section 5.2.3.1. Les résultats statistiques sur la structuration de cette scène sont présentés section 5.2.3.2. Ceux concernant la construction du graphe de visibilité sont introduits section 5.2.3.3.

5.2.3.1 Description de la scène

L'immeuble comprend 5 étages, chacun d'eux étant principalement composés de dizaines de pièces, de couloirs et corridors. Cet scène ne contient pas non plus d'objets et est dénuée de tout mobilier. Elle comporte en tout 8887 facettes dont 8687 sont rectangulaires et axiales. Le nombre total de positions axiales est de 1095. La figure 5.10 montre une vue en fil de fer de cette scène.

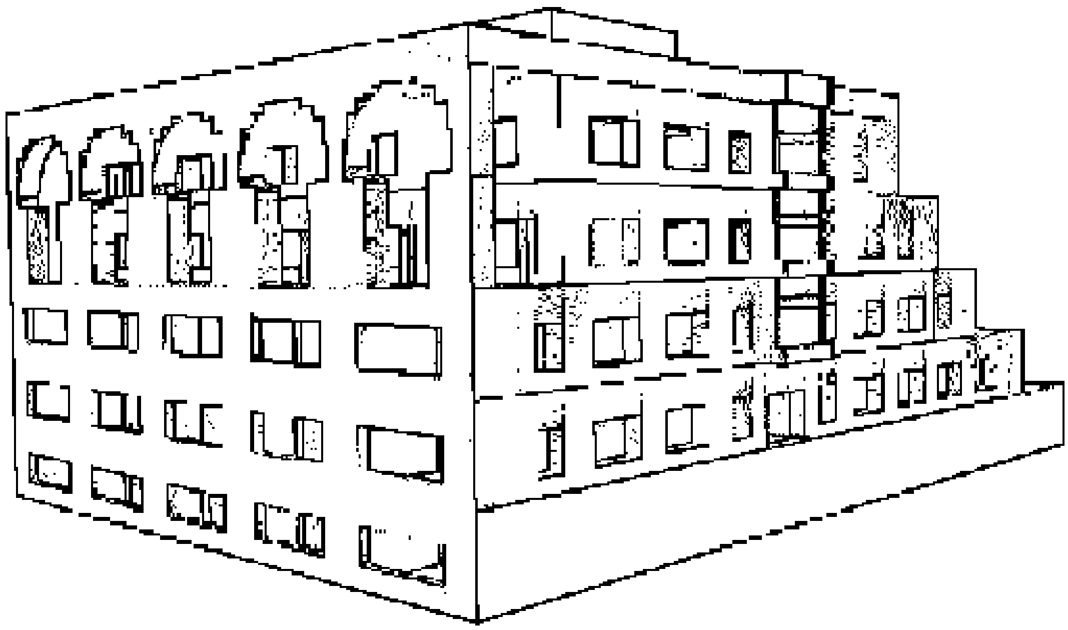

FIG. 5.10 - *Scène 3: l'immeuble*

5.2.3.2 Analyse de la subdivision

Nous présentons ici plusieurs mesures sur la structuration de la scène de *l'immeuble*. Ces mesures sont réparties dans trois tableaux: le tableau 5.9 regroupe les paramètres sur le découpage BSP, le tableau 5.10 regroupe ceux sur les cellules et le tableau 5.11 regroupe ceux sur les ouvertures. Ces résultats statistiques montrent des variations identiques à celles obtenues sur *l'étage*.

	Opacité = 0			**Opacité = 0.4**			**Opacité = 0.6**		
	C1	C2	C3	C1	C2	C3	C1	C2	C3
Nb de noeuds	29189	9743	8007	9237	7649	6463	1933	1939	1981
Nb de feuilles	14595	4872	4004	4619	3825	3232	967	970	991
Prof. maximum	22	40	48	30	40	47	36	36	36
Prof. moyenne	15	22	27	17	22	24	19	19	20
% faces supp.	175.1	38.6	21.7	40.6	32.0	23.1	12.0	12.0	12.3
% faces sur plans	98	97.4	97.7	91.3	83.2	79.2	36.4	36.2	36.6
Temps calcul BSP	4 s.	3 s.	352 s.	3 s.	2 s.	4 s.	2 s.	2 s.	2 s.

TAB. 5.9 - *Statistiques du découpage BSP de l'immeuble*

	Opacité = 0			**Opacité = 0.4**			**Opacité = 0.6**		
	C1	C2	C3	C1	C2	C3	C1	C2	C3
Nb cellules	14595	4872	4004	4619	3825	3232	967	970	991
% cell. vides	53.6	55.6	57.5	54.2	54.5	55.4	52.1	52.1	51.9
% cell. murales	45.9	43.9	42.1	45.5	45.0	44.2	46.9	47.0	46.9
% cell. peuplées	1.5	2.1	1.4	2.8	3.1	3.1	6.6	6.7	7.0
Temps calcul murs	2 s.	1 s.					< 1 s.		

TAB. 5.10 - *Statistiques des cellules de l'immeuble*

	Opacité = 0			**Opacité = 0.4**			**Opacité = 0.6**		
	C1	C2	C3	C1	C2	C3	C1	C2	C3
Nb ouvertures	42714	13465	11914	13206	10853	9803	4875	4883	4922
% vides-vides	58.1	68.6	70.4	52.9	52.2	50.6	20.7	20.7	20.8
% pleines-pleines	38.4	25.5	21.5	27.5	29.4	27.2	58.9	58.9	58.8
% pleines-vides	3.4	5.8	8.0	19.5	18.4	22.1	20.4	20.4	20.4
Temps calcul ouv.	822 s.	57 s.	42 s.	53 s.	32 s.	31 s.	5 s.	5 s.	5 s.

TAB. 5.11 - *Statistiques des ouvertures de l'immeuble*

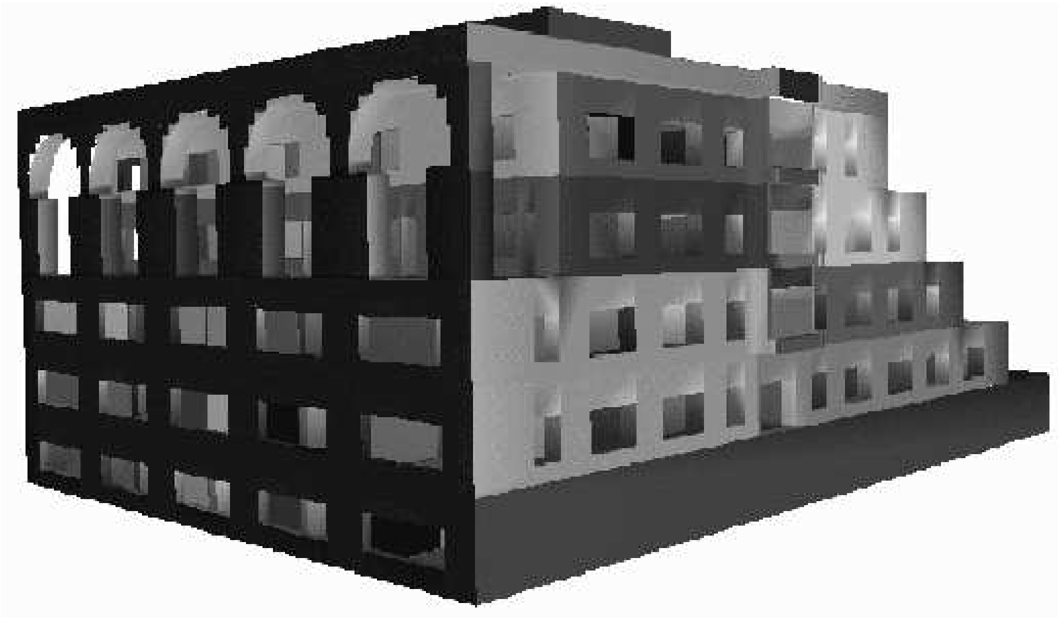

FIG. 5.11 - *Structuration obtenue sur l'immeuble*

La figure 5.11 montre une vue externe de *l'immeuble* après sa structuration en cellules. On distingue clairement les parties de la façade de droite appartenant à la même cellule. La façade de gauche constitue une unique cellule. Les critères de découpage basés sur l'aspect et le volume des cellules ont été désactivés. Nous pouvons ainsi facilement voir au travers des fenêtres de la façade les différentes cellules correspondantes aux pièces attenantes.

FIG. 5.12 - *Vue d'un couloir de l'immeuble. Il est découpé en plusieurs cellules dont les ouvertures apparaissent sur les côtés.*

5.2.3.3 Analyse du graphe de visibilité

Le tableau 5.12 présente les résultats concernant le calcul du graphe de visibilité sur *l'immeuble*.

	Opacité = 0			**Opacité = 0.4**			**Opacité = 0.6**		
	C1	C2	C3	C1	C2	C3	C1	C2	C3
Nb rayons ($\times 10^6$)	361.1	42.4	42.4	114.2	124.5	153.6	1458	1453.5	1437.8
Nb tests ($\times 10^5$)	417.4	47.2	46.8	124.1	131.4	161.9	1475.8	1471.2	1455.6
% réussis	19.7	13.8	13.3	11	7.0	7.1	1.6	1.6	1.6
Nb liens ($\times 10^4$)	103.6	11.9	8.6	15.3	10.1	7.9	3.5	3.6	3.8
Temps (en min.)	667	27	65	188	53	64	2055	625	905

TAB. 5.12 - *Statistiques de visibilité de l'immeuble*

Nous pouvons voir que le nombre de liens générés diminue quand le degré d'opacité minimum augmente. Cette variation est la même que celle relevée pour *l'étage*. Elle s'explique par une diminution du nombre d'ouvertures calculées dans la scène quand le degré d'opacité minimum augmente.

Pourtant, contrairement aux statistiques relevées pour les scènes *loft* et *étage*, les temps de calculs entre le second et le troisième groupe augmentent de façon importante. Ces temps s'expliquent par les nombres de tests inter-ouvertures effectués qui sont multipliés par 10 entre le second et le troisième groupe.

Pour comprendre la raison de cette augmentation du nombre de tests alors que le nombre d'ouvertures diminue, il faut se ramener à l'algorithme de construction du graphe de visibiilité.

Nous parcourons en profondeur les arcs du graphe d'adjacence, à partir d'un noeud donné. A chaque nouveau noeud rencontré, nous testons la visibilité de la cellule représentée par ce noeud avec la cellule du noeud de départ. Ces deux cellules ne peuvent être visibles qu'à travers la séquence des ouvertures traversées.

D'autre part, nous avons vu que les ouvertures traversées sont principalement des ouvertures reliant des cellules murales ou pleines. En effet, les cellules murales suivent les pièces et les embrasures des portes et fenêtres de la scène. Ainsi, en partant d'une cellule murale représentant une pièce, nous atteindrons par une première ouverture la cellule murale correspondant à une embrasure de porte. De même, depuis cette cellule murale nous atteindrons par une nouvelle ouverture la cellule murale représentant la pièce attenante.

Comme nous ne calculons les cellules visibles que depuis les cellules non vides (murales ou peuplées), nous parcourons donc uniquement les arcs représentant des ouvertures entre cellules pleines. Appelons *graphe de parcours* le sous-graphe du graphe d'adjacence ainsi parcouru.

	Opacité = 0.4			**Opacité = 0.6**		
	C1	C2	C3	C1	C2	C3
Nb ouv. / cell. pleine	2.93	2.98	3.36	8.35	7.68	8.16

TAB. 5.13 - *Nombres d'ouvertures par cellule pleine*

Le tableau 5.13 nous donne les nombres moyens d'ouvertures par cellule pleine (murale ou peuplée) dans le second et le troisième groupe. Notons que, seules les ouvertures reliant au moins une cellule pleine, sont prises en compte. Nous pouvons constater que les nombres du troisième groupe sont environ 2.5 fois supérieurs à ceux du second groupe. Cela signifie que le nombre d'arcs du graphe, partant d'un noeud représentant une cellule pleine, est 2.5 plus grand dans le troisième groupe que dans le second. Ce nombre d'arc partant d'un noeud est également appelé le degré d'un noeud.

Finalement, en augmentant le degré moyen des noeuds d'un graphe, le nombre de chemins possibles, à partir d'un noeud donné, augmente de façon importante. De ce fait, le nombre de tests de visibilité entre cellules augmente d'autant. Ce qui explique les fortes valeurs obtenues dans le tableau 5.12. De plus, comme le nombre de liens générés reste faible dans le troisième groupe (moins de 4×10^4 avec C2 et C3), le pourcentage de tests réussis devient du même coup extrèmement faible (1.6 %).

Ces dernières statistiques montrent les limites de la méthode utilisée pour établir la visibilité entre deux cellules. Ces limites sont principalement dues au fait que, quelques soient les cellules atteintes pendant le parcours du graphe d'adjacence, toutes leurs ouvertures sont testées pour continuer la progression. L'explosion combinatoire qui en résulte ralenti terriblement l'algorithme.

Or, à mesure que la séquence des ouvertures traversées s'allonge, il est évident que les parties réellement visibles des cellules atteintes sont de plus en plus réduites. La figure 5.7 illustre bien ce problème: seule une toute petite partie de la dernière cellule (la plus éloignée du point de vue) est visible.

En diminuant le nombre de chemins parcourus dans le graphe d'adjacence, nous réduisons donc le nombre de tests de visibilité inter-ouvertures. L'algorithme calculant le graphe de visibilité sera d'autant plus rapide que le nombre de tests de visibilité inter-ouvertures sera faible.

5.2.3.4 Influence du choix du plan de découpage

Nous illustrons ici l'influence du critère secondaire choisissant un plan de découpage parmi les plans candidats retenus par le critère d'opacité minimum, sur la structuration de l'immeuble.

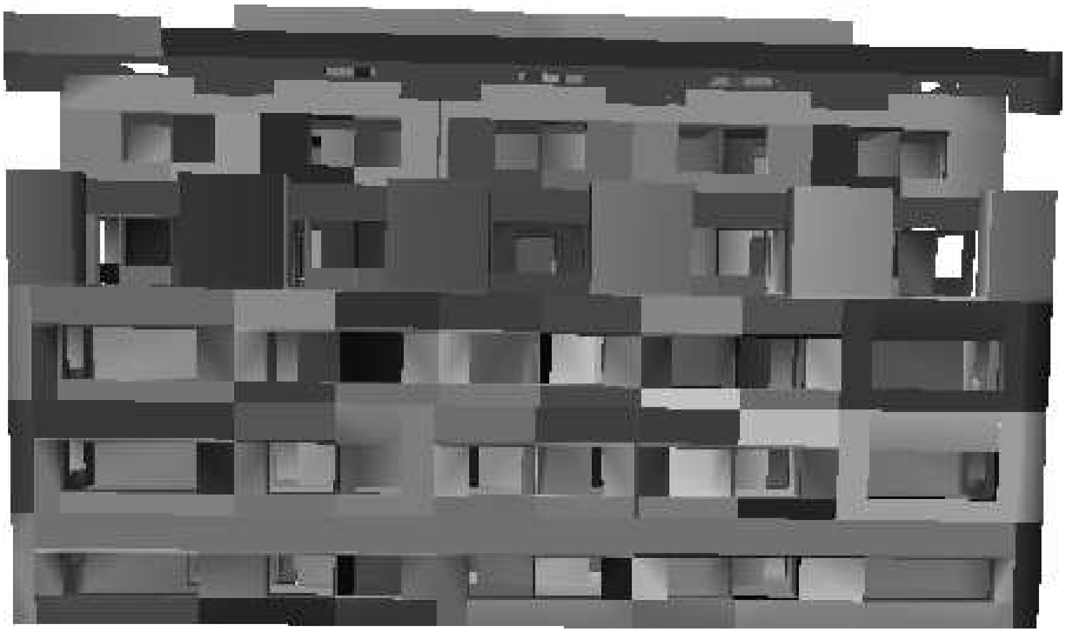

FIG. 5.13 - *Mauvais découpage de l'immeuble, vue d'une facade*

La figure 5.13 nous montre une structuration de l'immeuble en prenant le critère C1 (choix du plan médian). Nous constatons, d'une part, un nombre trop important de cellules, et d'autre part, un mauvais suivi de la topologie de la scène.

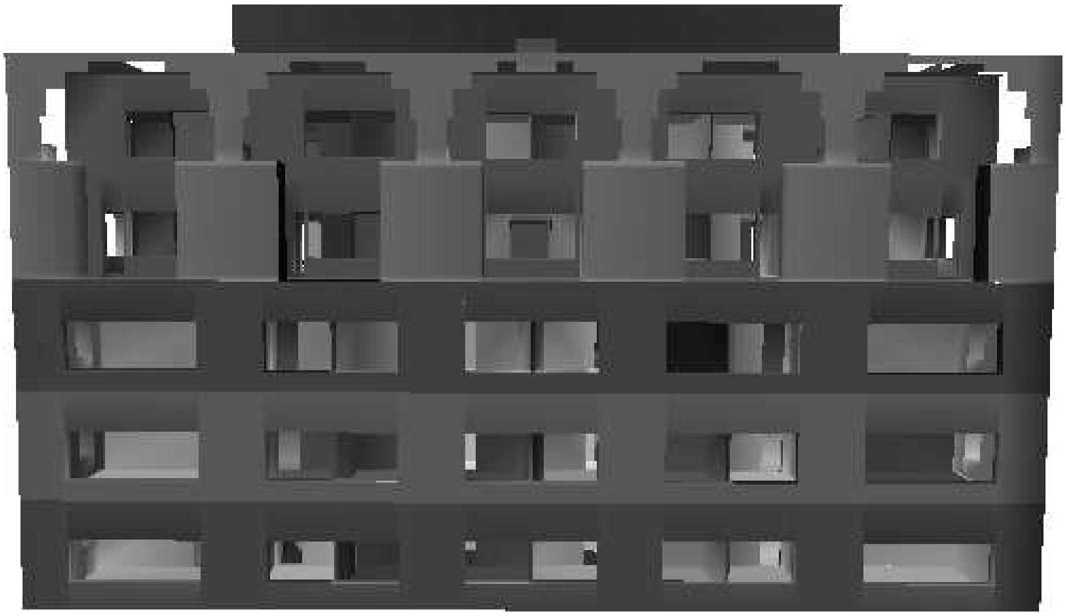

FIG. 5.14 - *Découpage moyen de l'immeuble, vue d'une facade*

La figure 5.14 nous montre ici une structuration de l'immeuble en prenant le critère C2 (choix du plan d'opacité maximum).

Le découpage obtenu est nettement meilleur que celui de la figure 5.13. La façade est découpée horizontalement en 5 cellules. Ceci s'explique par le fait qu'en prenant les plans d'opacité maximum, nous découpons prioritairement selon les plans correspondants aux sols et plafonds des étages de l'immeuble.

En revanche, certaines pièces sont encore découpées en plusieurs cellules. Les plans d'opacité maximum ne tiennent pas compte du nombre de coupures qu'ils provoquent.

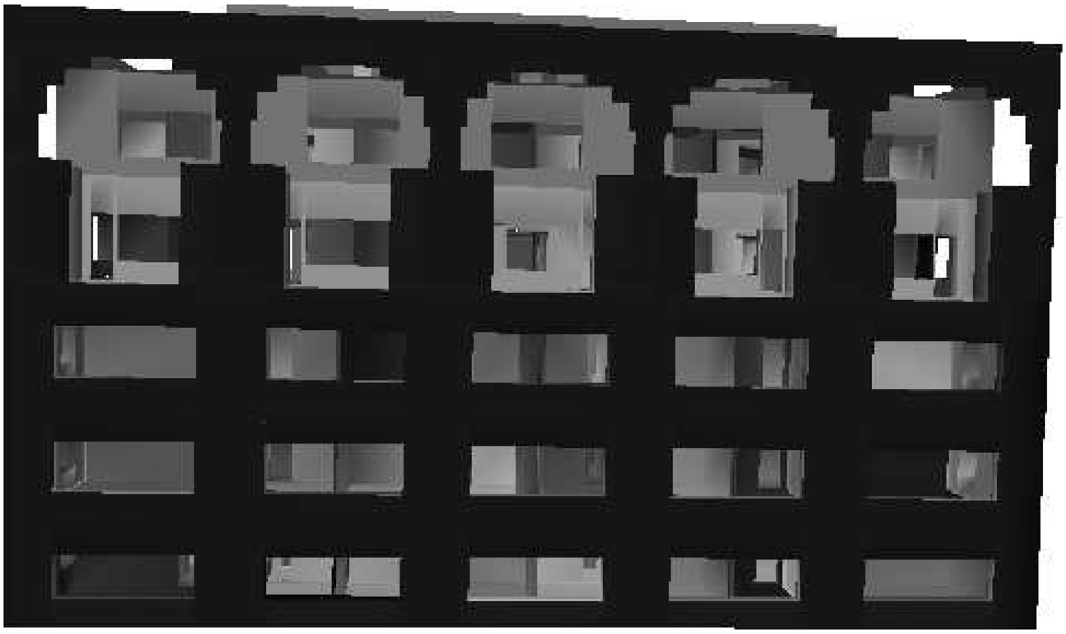

FIG. 5.15 - *Bon découpage de l'immeuble, vue d'une facade*

La figure 5.15 nous montre ici une structuration de l'immeuble en prenant le critère C3 (choix du plan découpant le moins de facettes).

Le découpage obtenu est encore amélioré par rapport à la figure 5.13. La façade n'est plus découpée. Ceci s'explique par le fait qu'en prenant les plans coupant le moins de facettes, les plans supportant les façades de l'immeuble sont choisis prioritairement.

5.3 Conclusion

Les différents résultats statistiques obtenus sur les trois scènes précédentes, montrent qu'il est possible d'obtenir une bonne structuration en jouant uniquement sur deux critères:

- **critère d'opacité**

  Le critère d'opacité permet de ne pas retenir les objets non occlusifs pour structurer la scène. Pour cela, la valeur minimum du degré d'opacité d'un plan de découpage ne doit pas être trop faible. De cette façon, seuls les éléments structurants majeurs d'une scène (tels que les murs, sols ou plafonds) seront choisis pour découper la scène. Le nombre de plans de découpage introduits et donc le nombre de cellules générées est alors réduit.

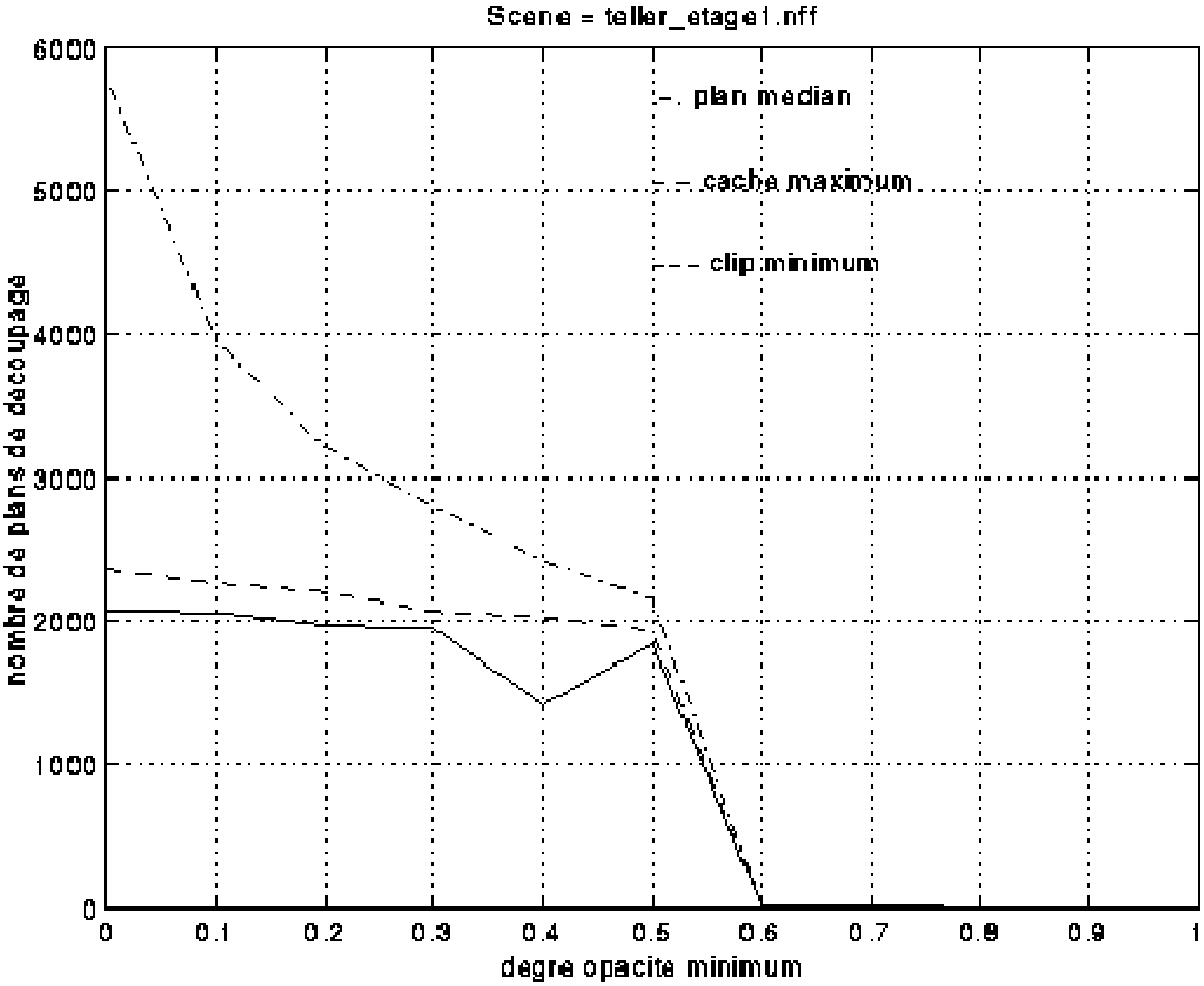

FIG. 5.16 - *Variation du nombre de plans de découpage en fonction du degré d'opacité minimum*

Cependant, en prenant un degré d'opacité trop fort, le risque est alors de ne plus découper du tout. Si nous considérons, par exemple, le découpage de *l'étage*, il existe un seuil pour l'opacité minimum au delà duquel il n'existe plus de plans candidats. La figure 5.16 représente la variation du nombre de plans de découpage introduits par l'algorithme en fonction du degré d'opacité minimum. Une courbe pour chaque critère secondaire choisi (C1, C2 et C3) est tracée. Nous pouvons ainsi remarquer qu'en prenant des valeurs supérieures à 0.5 pour le degré d'opacité minimum, et quelque soit le critère secondaire choisi, il n'y a plus de plans candidats pour découper la scène.

- **critère secondaire**

  L'importance du critère secondaire sur le nombre de plans de découpage apparaît également nettement sur la figure 5.16. Ainsi, les critères C2 et C3 maintiennent ce nombre à une valeur relativement constante par rapport au critère C1. La raison est que le critère C1 n'étudie pas l'influence des différents plans de découpage candidats sur la structuration de la scène. Son seul but est de réduire le déséquilibrage de l'arbre. En revanche, C2 et C3 s'intéressent directement aux conséquences du choix des plans de découpage sur les cellules. Alors que C2 choisit systématiquement le plan d'opacité maximum, C3 cherche à limiter le nombre de coupures provoquées sur les facettes. Les meilleurs résultats sont obtenus avec C3: moins de plans de découpages, moins de cellules, et donc moins d'ouvertures.

  Les figures 5.17 et 5.18 illustrent ainsi l'avantage d'utiliser le critère C3 pour limiter le pourcentage d'augmentation du nombre de facettes et le nombre d'ouvertures.

Enfin, nous avons vu que les temps de calcul du graphe de visibilité sont loin d'être négligeables dès que la scène comporte plusieurs milliers de facettes. Ici, c'est le nombre d'ouvertures générées et leur répartition (% ouvertures entre cellules pleines, entre cellules vides, et entre cellules pleines et cellules vides) qui influeront sur ces temps de calcul. Moins il y a d'ouvertures entre cellules pleines, plus les calculs seront rapides.

La figure 5.19 représente la variation du temps de calcul du graphe de visibilité en fonction du degré d'opacité minimum. Ici, c'est le critère C2 qui permet d'obtenir les meilleurs temps de calcul (c'est à dire les plus courts). Ce résultat n'est pas surprenant puisque le critère C2 choisit justement les plans de découpage de manière à réduire au maximum la visibilité des cellules entre-elles.

Notons enfin, que les pourcentages des tests de visibilité inter-ouvertures réussis restent faible en raison du nombre de chemins bien trop important empruntés dans le graphe d'adjacence. Un travail d'optimisation de l'algorithme calculant le graphe de visibilité reste donc à faire.

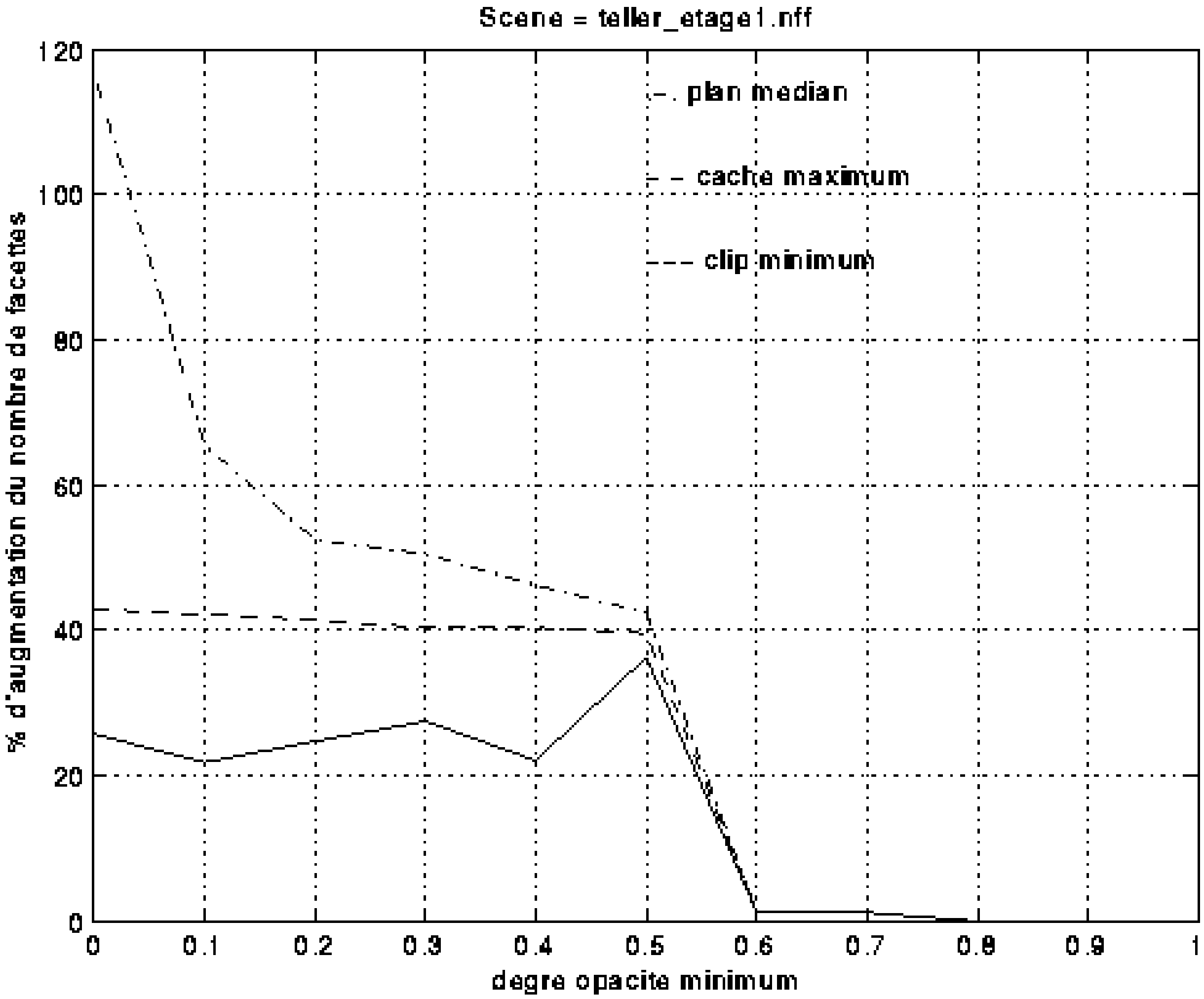

FIG. 5.17 - *Variation du pourcentage d'augmentation du nombre de facettes en fonction du degré d'opacité minimum*

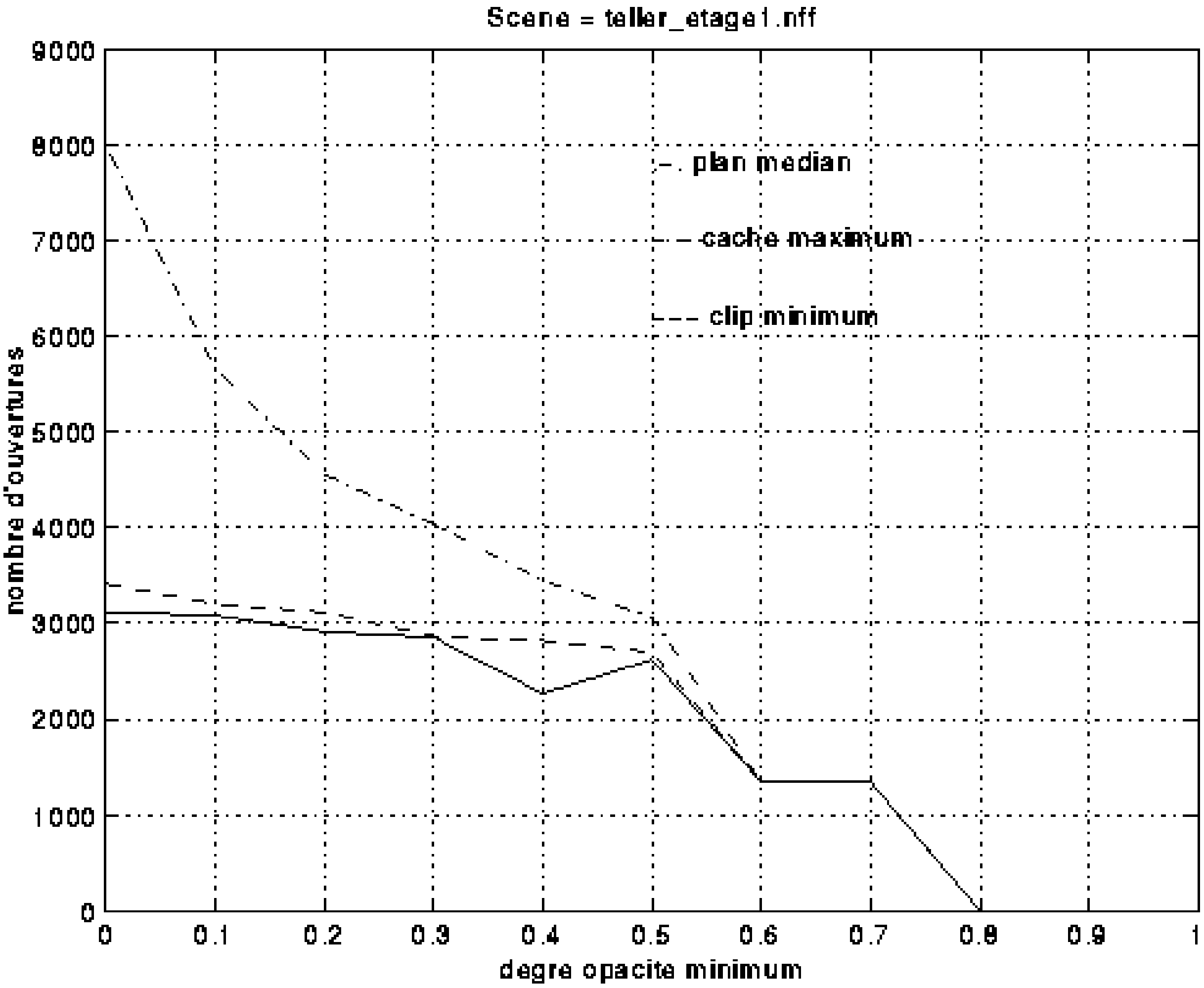

FIG. 5.18 - *Variation du nombre d'ouvertures en fonction du degré d'opacité minimum*

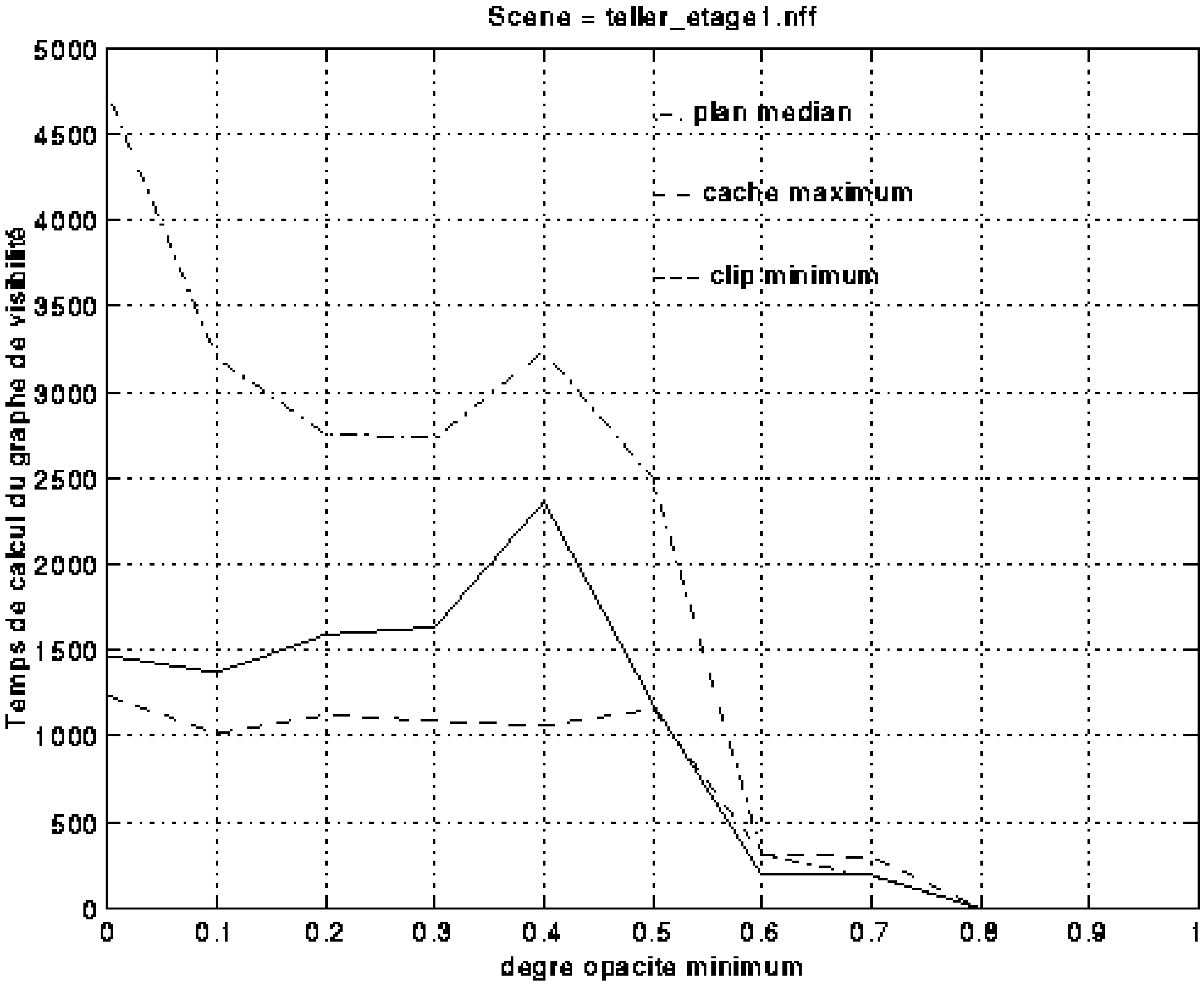

FIG. 5.19 - *Variation du temps de calcul du graphe de visibilité en fonction du degré d'opacité minimum*

Chapitre 6

Conclusion

Dans cette étude, nous nous sommes intéressés à diminuer la complexité du calcul des visibilités sur des scènes axiales. Ces travaux intéressent, en particulier, le rendu photoréaliste de scènes complexes ainsi que leur visualisation interactive.

Notre approche consiste à précalculer certaines relations de visibilité entre les objets d'une scène. Pour cela, nous proposons deux étapes principales: structurer la scène en cellules et ouvertures, puis calculer un graphe de visibilité entre ces cellules. L'étape de structuration a été particulièrement approfondie pour examiner les différents paramètres conduisant rapidement à un graphe de visibilité performant.

Pour structurer la scène, nous utilisons un découpage de la scène à l'aide de plans de partition binaire. Le critère d'opacité, déjà introduit par Airey, permet ainsi de retenir plusieurs plans de découpage. Cependant, ce critère reste insuffisant pour obtenir une structuration efficace de la scène. Nous avons donc rajouter trois critères secondaires différents afin d'obtenir une meilleure structuration.

Les résultats obtenus par notre algorithme montrent que le critère d'opacité, couplé au critère de réduction du nombre de coupures, donne une bonne structuration de la scène: les cellules obtenues suivent de près les pièces et couloirs existants dans la scène.

Dans un souci d'améliorer les performances de notre algorithme de structuration, nous avons également apporté des solutions aux problèmes posés par les cellules trop grandes, trop allongées ou encore non convexes.

En tirant parti de cette structuration, nous construisons le graphe de visibilité entre les cellules. Ce graphe indique pour chaque cellule les cellules qui lui sont visibles. Les différents fichiers générés par notre algorithme permettent de représenter facilement ce graphe.

En utilisant ce graphe de visibilité pour traiter des modèles géométriques complexes, il est alors possible d'accélérer les calculs d'illumination globale, en ne chargeant en mémoire que les cellules participant aux échanges lumineux.

De même, ce graphe de visibilité permet de visualiser rapidement la traversée d'environnements complexes. Ainsi, seules les cellules visibles de la cellule occupée par l'observateur peuvent être chargées en mémoire pour visualiser la scène.

Malgré cela, plusieurs problèmes restent à résoudre.

Ainsi, l'algorithme calculant le graphe de visibilité est peu efficace dès que le nombre de cellules augmente.

La gestion du graphe de visibilité reste également à étudier pour optimiser les échanges entre la mémoire et la base de données regroupant les cellules d'une scène. Une parallélisation de cette gestion du graphe permettrait certainement de réduire ces coûts de communication.

Enfin, le passage aux scènes non axiales permettrait de traiter des scènes plus générales.

Annexes

Annexe A

Fichier de configuration

Pour des raisons pratiques, toutes les options de notre programme doivent être précisées dans un fichier de configuration. Lorsque le programme DEA est lancé, il recherche le fichier de configuration *dea.cfg* dans le répertoire courant. Pour préciser un autre fichier de configuration, par exemple *loft.cfg*, il suffit de taper: *dea -i loft.cfg*. Si le fichier de configuration n'est pas trouvé, le programme s'arrête.

Cette annexe décrit la syntaxe des options disponibles dans ce fichier. Le fichier de configuration est un fichier texte. Toute ligne vide ou commençant par '#' est ignorée. Une seule option par ligne est autorisée. Le format pour définir la valeur d'un champ est:

NomChamp = *ValeurOption*

Seul le champ **FichierScene** est obligatoire. Si un champ est défini plusieurs fois, c'est la dernière valeur qui est prise en compte.

A.1 Mode de fonctionnement général

Le champ **ModeProgramme** indique le mode d'éxécution du programme. Deux valeurs sont possibles:

- *NORMAL*

  Indique que le programme doit s'éxécuter en mode normal: un seul découpage de la scène est effectué.

- *STAT*

  Indique que le programme doit s'exécuter en mode statistique: plusieurs découpages sont effectués et plusieurs fichiers statistiques sont produits. Les paramètres mesurés portent sur les arbres BSP générés et les cellules obtenues. Pas de calculs de visibilité. Pour plus d'informations, se reporter au chapitre 5.

NORMAL est la valeur par défaut.

A.2 Fichiers manipulés

Ces options concernent les noms et type de fichiers manipulés par le programme.

A.2.1 Nom de la scène

Le champ **FichierScene** indique le nom du fichier scène à lire. Le fichier doit être au format NFF. Un chemin d'accès peut être précisé devant le nom. Ce champ est obligatoire et doit être non vide.

Exemple:

FichierScene = *Scenes/Loft.nff*

A.2.2 Graphe de visibilité

Le champ **FichierGraphe** indique le nom du fichier décrivant le graphe de visibilite des cellules. Le format de ce fichier est détaillé section 3.4.3. Si le champ est vide, le fichier du graphe de visibilité n'est pas écrit.

Exemple:

FichierGraphe = *Graphes/Loft.gra*

A.2.3 Contrôle global

Le champ **FichierControleGlobal** indique le nom du fichier de contrôle global de la décomposition. Ce fichier décrit, dans le format RAD, les facettes issue de la décomposition de la scene. Les facettes appartenant a une même cellule ont la même couleur. Si le champ est vide, le fichier de contrôle global n'est pas écrit.

Exemple:

FichierControleGlobal = *Rad/Loft.rad*

A.2.4 Contrôle de visibilité

Le champ **NomGenControleVisibilite** indique le nom générique des fichiers de contrôle visibilité cellule. Un fichier de contrôle visibilité cellule est un fichier de contrôle global où:

1. Les facettes de la cellule controlée sont écrites en rouge.
2. Les facettes des cellules visibles sont écrites en dégrade de vert (du vert clair au vert foncé selon la distance de cette cellule à la cellule controlée).
3. Les facettes des autres cellules sont écrites dans des couleurs différentes en respectant cependant une même couleur par cellule.

Un nom de fichier de contrôle visibilité sera la concaténation du nom générique et du numéro de la cellule. Si le champ est vide ou que les calculs de visibilités ne sont pas faits, les fichiers de contrôle visibilité cellule ne seront pas écrits.

Exemple:

NomGenControleVisibilite = *Controles/Loft/Loft_ctrl_cel_*

A.2.5 Statistiques

Le champ **NomGenFichierStatistique** indique le nom générique des fichiers recueillant les divers paramètres mesurés si le programme est lancé en mode statistique. Ce nom générique est également utilisé pour le nom des fichiers de commandes MatLab affichant ces statistiques. Si le champ est vide, les sorties se feront sur la sortie standard.

Exemple:

NomGenFichierStatistique = *Stat/Loft/s*

Si 14 paramètres sont mesurés, les fichiers de commandes MatLab *s1.m*, *s2.m*, ..., *s14.m* ainsi que les fichiers contenant les valeurs statistiques mesurées, seront écrits dans le répertoire *Stat/Loft*.

A.2.6 Cellules

Le champ **NomGenCellule** indique le nom générique des fichiers décrivant les cellules générées. Un nom de fichier cellule sera la concaténation du nom générique et du numéro de la cellule. Un fichier cellule décrit les facettes appartenant à la cellule (facettes murales et incluses). Le format de sortie de ce fichier est indiqué par le champ **FormatCellule**. Si le champ est vide, les fichiers cellules ne seront pas écrits.

Exemple:

NomGenCellule = *Cellules/Loft/Loft_cel_*

A.2.7 Format des cellules

Le champ **FormatCellule** indique le format de sortie des fichiers cellules. Les valeurs autorisees sont *RAD* pour le format RAD, et *NFF* pour le format NFF. La valeur par défaut est *NFF* si le champ est vide.

Exemple:

FormatCellule = *NFF*

A.3 Options de découpage

Ces champs concernent le découpage de la cellule et la valeur des critères de sélection des plans de découpage.

A.3.1 BSP seulement

Le champ **BspSeul** indique si seul le découpage BSP de la scène sera effectué. *ON* active le champ (pas de calculs de visibilité), *OFF* désactive le champ. La valeur par défaut est *OFF*.

Exemple:

BspSeul = *OFF* autorise des calculs de visibilité.

A.3.2 Orientation des plans

Le champ **DecoupX** (resp. **DecoupY**, **DecoupZ**) indique si des plans de découpage perpendiculaires à l'axe des X (resp. Y, Z) sont autorisés. *ON* active le champ, *OFF* désactive le champ. La valeur par défaut est *ON*.

Exemple:

DecoupX = *ON* autorise des plans de découpage perpendiculaires à l'axe des X.

A.3.3 Critère de sélection secondaire

Le champ **PlanSelection** indique le critère de sélection d'un plan de découpage si plusieurs plans sont candidats à l'issue du critère global (opacité minimum.). Les valeurs possibles sont :

- *MEDIAN* pour choisir le plan médian.
- *CACHE_MAX* pour choisir le plan d'opacité maximum.
- *CLIP_MIN* pour choisir le plan découpant le moins de facettes.

La valeur par défaut est *CLIP_MIN*.

Exemple:

PlanSelection = *CLIP_MIN*

A.3.4 Opacité minimum

Le champ **OpaciteMin** indique le seuil d'opacité minimum (critère principal) pour qu'un plan de découpage soit retenu. En mode *STAT*, ce champ indique la première valeur du seuil d'opacité minimum. Les valeurs autorisées sont tous les réels de l'intervalle $]0, 1.0]$. La valeur par défaut est 0.5.

Exemple:

OpaciteMin = 0.4 signifie une opacité minimum de 40% pour sélectionner un plan découpage.

A.3.5 Dernier seuil opacité

Le champ **OpaciteMax** indique la dernière valeur du seuil d'opacité minimum dans une étude statistique. Cette valeur doit être $\geq$ **OpaciteMin** mais toujours ≤ 1. La valeur par défaut est 1.0. Ce champ n'est valable que pour le mode d'éxécution *STAT*, il n'est pas pris en compte si le programme s'exécute en mode *NORMAL*.

Exemple:

OpaciteMax = 1.0

A.3.6 Incrément du seuil d'opacité

Le champ **OpacitePas** indique le pas d'incrémentation pour le seuil minimum d'opacité. Cette valeur doit être un réel de l'intervalle $]0, 1.0[$. La valeur par défaut est 0.01. Ce champ n'est valable que pour le mode d'éxécution *STAT*, il n'est pas pris en compte si le programme s'exécute en mode *NORMAL*.

Exemple:

OpacitePas = 0.01

A.3.7 Volume maximum d'une cellule

Le champ **VolumeMax** indique le rapport maximum autorisé entre le volume d'une cellule et celui de la scène. Si ce rapport est dépassé, la cellule est découpée. Cette valeur doit être un réel de l'intervalle $]0, 1.0]$. La valeur par défaut est 0.03.

Exemple:

VolumeMax = 0.04 signifie qu'une cellule ne doit pas avoir un volume supérieur à 4% de celui de la scène.

A.3.8 Aspect minimum d'une cellule

Le champ **AspectMin** indique l'aspect minimum autorisé pour une cellule. Si un aspect d'une cellule est plus petit que cette valeur, la cellule est découpée. Cette valeur doit être un réel de l'intervalle $]0, 1.0[$. La valeur par défaut est 0.01.

Exemple:

AspectMin = 0.02 signifie qu'une cellule ne doit pas avoir un côté 5 fois $(1 \div 0.2)$ plus long qu'un autre.

A.4 Options pour les calculs de visibilité

Ces options concernent les calculs de visibilité entre cellule.

A.4.1 Nombre de rayons

Le champ **RayonsMax** indique le nombre maximum de rayons à lancer pour tester la visibilité entre deux ouvertures. Les valeurs autorisées sont les entiers ≥ 1. La valeur par défaut est 10.

Exemple:

RayonsMax = 10

A.4.2 Objets opaques

Le champ **CacheObjets** indique si les facettes incluses dans les cellules (différentes des facettes murales) sont ignorées dans les calculs de visibilité entre cellules. *ON* active le champ, *OFF* désactive le champ. La valeur par défaut est *ON*.

Exemple:

CacheObjets = *ON* cache les facettes incluses dans les cellules.

A.5 Options d'affichage

Ces options concernent les messages et statistiques affichés par le programme.

A.5.1 Avertissements et repères d'exécution

Le champ **ModeQuiet** indique si les avertissements et les repères d'exécution ne doivent pas être affichés. Des avertissments sont par exemple générés si des facettes non axiales ou non rectangulaires sont trouvées (ces facettes ne sont cependant pas éliminées).

Les repères d'exécution sont les messages délimitant les principales étapes du programme en affichant également leurs temps d'exécution.

ON active le champ, *OFF* désactive le champ. La valeur par défaut est *ON*.

Exemple:

ModeQuiet = *ON* active l'affichage des avertissements et des repères d'exécution.

A.5.2 Statistiques de la scène

Le champ **StatistiquesScene** indique si des statistiques sur la scène lue doivent être affichées. Les informations affichées concernent par exemple le nombre de facettes, leurs positions et le degré d'opacité des plans de découpage de la première cellule.

ON active le champ, *OFF* désactive le champ. La valeur par défaut est *ON*.

Exemple:

StatistiquesScene = *OFF* désactive l'affichage.

A.5.3 Statistiques de l'arbre BSP

Le champ **StatistiquesBSP** indique si des statistiques sur l'arbre BSP obtenu doivent être affichées. Les informations affichées concernent par exemple le nombre de noeuds, l'équilibrage de l'arbre, ou encore le nombre de facettes présentes sur les plans de découpage.
ON active le champ, *OFF* désactive le champ. La valeur par défaut est *ON*.

Exemple:

StatistiquesBSP = *OFF* désactive l'affichage de ces informations.

A.5.4 Statistiques des cellules

Le champ **StatistiquesCellules** indique si des statistiques sur les cellules obtenues doivent être affichées. Les informations affichées concernent par exemple le nombre de cellules obtenus, le pourcentage de cellules vides, ou encore leurs volumes et aspects.
ON active le champ, *OFF* désactive le champ. La valeur par défaut est *ON*.

Exemple:

StatistiquesCellules = *OFF* désactive l'affichage de ces informations.

A.5.5 Statistiques de visibilité

Le champ **StatistiquesVisibilite** indique si des statistiques concernant les calculs de visibilité effectués doivent être affichées. Les informations affichées concernent par exemple le nombre de rayons de visibilité total lancés, le nombre moyen de rayons lancés par test de visibilité entre deux ouvertures, ou encore le nombre de liens de visibilité générés.
ON active le champ, *OFF* désactive le champ. La valeur par défaut est *ON*.

Exemple:

StatistiquesVisibilite = *OFF*

Annexe B

Formats des fichiers générés

Plusieurs fichiers sont générés par notre programme. Le mode d'exécution (*NORMAL* ou *STAT*) ainsi que les options choisies dans le fichier de configuration (voir annexe A) déterminent les différents types de fichiers générés. Si le programme est lancé en mode *NORMAL*, seuls les fichiers cellules, le graphe de visibilité et les fichiers de contrôle peuvent être écrits. Les formats respectifs de ces fichiers sont présentés section B.1, B.2 et B.3.

Si le programme s'exécute en mode *STAT*, seuls les fichiers statistiques et de commandes MatLab sont générés. Les formats respectifs de ces fichiers sont présentés section B.4 et B.5.

B.1 Format des fichiers cellules

Un fichier cellule regroupe les facettes appartenant à la même cellule. Ce fichier décrit uniquement la géométrie de ces facettes, c'est à dire les coordonnées de leurs sommets. Deux formats peuvent être utilisés: soit NFF, soit RAD. Ce choix est laissé à l'utilisateur à l'aide de l'option **FormatCellule** dans le fichier de configuration (voir annexe A). Par défaut, c'est le format NFF qui est utilisé.

La première ligne du fichier cellule indique le format choisi en le précédant du caractère # (#NFF ou #RAD). Les lignes suivantes dans le fichier diffèrent selon le format de sortie.

Pour les fichiers cellules au format RAD:

- nombre de facettes dans le fichier;
- pour chaque facette:
  - le nombre de sommets;
  - pour chaque sommet, ses trois coordonnées séparées par un espace.

Pour les fichiers cellules au format NFF, pour chaque facette:

- le caractère 'p', un espace, et le nombre de sommets;
- pour chaque sommet, ses trois coordonnées séparées par un espace.

Enfin, le nom d'un fichier cellule est obtenu en concaténant le nom générique précisé dans le fichier de configuration (option **NomGenCellule**) avec le numéro de la cellule.

B.2 Format du fichier du graphe de visibilité

Le nom du fichier décrivant le graphe de visibilité des cellules est indiqué par l'option **FichierGraphe** dans le fichier de configuration. Ce fichier doit contenir les noeuds et les arcs du graphe. Un noeud représente une cellule non vide. Un arc relie deux cellules visibles.

Le fichier contient donc les lignes suivantes:

- le nombre de noeud du graphe (i.e. le nombre de cellules non vides);
- puis pour chaque noeud, et sur la même ligne:
  - le numéro de la cellule représentée;
  - le nom du fichier cellule associé.
- le nombre d'arc du graphe;
- puis pour chaque arc, et sur la même ligne:
  - le numéro de la cellule du premier noeud de l'arc;
  - le numéro de la cellule du second noeud de l'arc.

Notons que la visibilité de deux cellules entre elles est représentée par un seul arc entre deux noeuds.

B.3 Format des fichiers de contrôle

Deux types de fichier de contrôle permettent de contrôler la structuration d'une part, et la visibilité des cellules d'autre part. Les options **FichierControleGlobal** et **NomGenControleVisibilite** du fichier de configuration indiquent les noms de ces fichiers. L'annexe A décrit plus précisément le contenu de ces fichiers. Leur format est cependant le même: il s'agit du format RAD.

Le format est le suivant:

- le nombre de facettes dans le fichier;

- puis pour chaque facette:
  - le nombre de sommets;
  - pour chaque sommet, et sur la même ligne:
    - ses trois coordonnées séparées par un espace;
    - ses trois composantes RGB séparées par un espace;

Les composantes RGB définissent une couleur. Une facette ayant une couleur uniforme, les couleurs de ses sommets seront les mêmes.

B.4 Format des fichiers statistiques

Les fichiers statistiques mesurent plusieurs paramètres sur la structuration et le graphe de visibilité obtenus. Le champ **NomGenFichierStatistique** du fichier de configuration permet d'indiquer un nom générique pour ces fichiers. Des fichiers MatLab automatiquement générés (voir section suivante) permettent d'obtenir les courbes statistiques correspondantes.

Exceptées les statistiques concernant le volume des cellules, le format de ces fichiers est le même. En effet, un fichier représente la variation d'un paramètre donné en fonction du degré d'opacité minimum et pour un critère secondaire donné.

Pour un paramètre, nous obtenons donc trois fichiers correspondants aux trois critères secondaires.

Soit X le paramètre mesuré, le fichier statistique correspondant comprend autant de lignes que de valeurs mesurées (i.e. de valeurs différentes du degré d'opacité minimum). Chaque ligne est constituée du couple de valeurs suivantes:

- le degré d'opacité minimum;
- la valeur de X pour ce degré d'opacité.

Si N est le nombre de paramètres mesurés en fonction du degré d'opacité minimum, $3 \times N$ fichiers statistiques seront générés.

En ce qui concerne le volume des cellules, nous voulons distinguer les cellules vides des cellules pleines. Pour un degré d'opacité minimum et pour un critère secondaire donné nous générons un fichier contenant les volumes des cellules vides et un fichier contenant les volumes des cellules pleines. Les volumes sont écrits dans le fichier sous la forme d'une suite de valeurs séparées par des espaces.

Si M est le nombre de valeurs prises par le degré d'opacité minimum, alors $2 \times 3 \times M$ fichiers, contenant chacun une série de volumes, seront générés.

B.5 Format des fichiers MatLab

MatLab permet de visualiser rapidement tout ensemble de valeurs numériques sous forme de courbes. De plus, un langage de commandes intégré permet de créer facilement de nouvelles fonctions sous la forme de fichiers de commandes texte.

Pour visualiser, sous forme de courbes, les différents paramètres statistiques calculés par notre programme, nous générons de tels fichiers. Ces fichiers ont obligatoirement l'extension '**.m**' pour être reconnu par MatLab, et sont placés dans le même répertoire que les fichiers statistiques.

Finalement, nous générons autant de fichiers de commandes que de paramètres mesurés.

Une barre de menus déroulants, sous MatLab, a été écrite pour les exécuter facilement sur toutes les scènes traitées. Ce menu est décrit section C.3.

Annexe C

Utilitaires

Pour pouvoir traiter plusieurs scènes par notre programme, nous avons écris deux utilitaires. Le premier permet de convertir des fichiers au format UNIGRAFIX vers le formatNFF. La scène de l'*immeuble* était initialement sous le format UNIGRAFIX. Cet utilitaire est présenté section C.1.

Le second utilitaire, présenté section C.2, permet d'extraire une partie d'une scène donnée. Il nous a ainsi permis d'extraire chaque étage de l'immeuble pour constituer de nouvelles scènes.

La section C.3 présente l'interface, écrit sous MATLAB, permettant d'afficher facilement les courbes statistiques de toutes les scènes.

C.1 Convertir de l'UNIGRAFIX en NFF

Le programme UG2NFF permet de convertir les facettes d'une scène du format UNIGRAFIX vers le format NFF. La conversion ne s'effectue qu'au niveau de la géométrie des facettes. Les données photométriques ne sont pas converties. La syntaxe d'appel est la suivante:

```
ug2nff -i[nput] fich [options]
```

Les options disponibles sont:

```
  -o[utput] nomfic : fichier de sortie (par défaut output.nff)
-f[acettes] : écrit uniquement les facettes en sortie
```

L'option -f ne recopie pas les lignes non interprétées du fichier d'entrée (commentaires, données photométriques, etc.) dans le fichier de sortie.

C.2 Extraire une sous-scène

Le programme DataClip permet d'extraire une sous-scène d'un fichier. Pour cela, une boite axiale de clipping est appliquée sur la scène. Seules les facettes contenues dans cette boite (certaines pouvant être coupées par les côtés de la boite) sont écrites dans le fichier de sortie.

Ce programme permet, par exemple, d'extraire individuellement les étages d'un immeuble. Il suffit de connaître les positions verticales du sol et du plafond de l'étage voulu.

La syntaxe d'appel est la suivante:

```
dc -i[nput] fich [options]
```

Les options disponibles sont:

```
-o[utput] nomfic : fichier de sortie (par défaut output)
-u[nigrafix] : fichier scène au format UniGrafix
-n[ff] : fichier scène au format NFF (par défaut)
-x[min] val : xmin de l'englobant de clipping
-y[min] val : ymin de l'englobant de clipping
-z[min] val : zmin de l'englobant de clipping
-X[max] val : xmax de l'englobant de clipping
-Y[max] val : ymax de l'englobant de clipping
-Z[max] val : zmax de l'englobant de clipping
```

Le format du fichier de sortie est le même que celui du fichier d'entrée. Les coordonnées xmin, ymin, zmin et xmax, ymax, zmax désignent les coordonnées des deux sommets extrêmes de la boite axiale de clipping.

C.3 Visualiser les courbes statistiques

Pour visualiser facilement la variation des différents paramètres mesurés, des fichiers de commandes MatLab sont automatiquement générés par le programme. Une barre de menus déroulante a été écrite sous MatLab pour permettre de les exécuter intéractivement. Cette barre de menus ne peut s'appeler que depuis l'environnement de MatLab en tapant la commande **dea**.

Le premier menu, **Scènes**, permet de sélectionner la scène désirée. Le *loft*, l'*étage* et l'*immeuble* peuvent ainsi être choisis. Les autres étages de l'immeuble ont également été traités, et les statistiques les concernant sont également disponibles dans ce menu.

Le second menu, **Courbes**, permet de sélectionner le paramètre statistique de son choix. Au total, 22 paramètres répartis en 4 groupes (arbre BSP, cellules, ouvertures et visibilité) sont disponibles. A chaque paramètre sélectionné, sa variation en fonction du degré d'opacité minimum est affichée. Comme nous voulons également comparer les trois critères secondaires entre eux, trois courbes sont finalement affichées (une par critère secondaire).

Le troisième menu, **Imprime**, permet d'imprimer les courbes affichées, ou encore toutes les courbes correspondantes à la scène courante.

Bibliographie

[1] K. Bouatouch & M. Madani & T. Priol & B. Arnaldi. A new algorithm of space tracing using a csg model. Eurographics'88, 1988.

[2] A. S. Glassner. Space subdivision for fast ray tracing. *IEEE Computer Graphics and Applications*, 4(10):15–22, October 1984.

[3] Jean-Philippe Thirion. Tries : data structures based on binary representation for ray tracing. In C.E. Vandoni and D.A. Duce, editors, *Eurographics '90*, pages 531–541, Eurographics Association, Elsevier Science Publishers, 1990.

[4] C. B. Jones. A new approach to the 'hidden line' problem. *The Computer Journal*, 14(3):232–271, March 1971.

[5] Frederick P. Brooks John M. Airey, John H. Rohlf. Towards image realism with interactive update rates in complex virtual building environments. *ACM*, 41–50, May 1990.

[6] Harry Plantiga & Charles R. Dyer. Visibility, occlusion, and the aspect graph. *International Journal of Computer Vision*, 5(2):137–160, 1990.

[7] Harry Plantiga. Conservative visibility preprocessing for efficient walkthroughs of 3d scenes. *Graphics Interface '93*, 166–173, 1993.

[8] Seth J. Teller & Carlo H. Sequin. Visibility preprocessing for interactive walkthroughs. *Computer Graphics*, 25(4):61–69, July 1991.

[9] Seth Jared Teller. *Visibility Computations in Density Occluded Polyhedral Environments*. PhD thesis, University of California at Berkeley, 1992.

[10] Seth Teller & Pat Hanrahan. Global visibility algorithms for illumination computations. In *Computer Graphics Proceedings, Annual Conference Series*, pages 239–246, 1993.

[11] Seth Teller & Celeste Fowler & Thomas Funkhouser & Pat Hanrahan. Partitioning and ordering large radiosity computations. In *Computer Graphics Proceedings, Annual Conference Series*, pages 443–450, 1994.

[12] Pat Hanrahan & David Salzman & Larry Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics*, 25(4):197–205, July 1991.

[13] Chris Georges David Luebke. Portals and mirrors: simple, fast evaluation of potentially visible sets. *Symposium on Interactive 3D Graphics*, 105–106, 1995.

[14] M. Gilliland & W. Sharp R. A. Schumacker, R. Brand. *Study for Applying Computer-Generated Images to Visual Simulation.* AFHRL-TR-69-14, U.S. Air Force Human Resources Laboratory, 1969.

[15] H. Fuchs & Z. Kedem & B. Naylor. On visible surface generation by a priori tree structures. In *Computer Graphics*, pages 124–133, March 1980.

[16] William Thibault Bruce Naylor, John Amanatides. Merging bsp trees yields polyhedral set operations. *Computer Graphics*, 24(4):115–124, August 1990.

[17] Norman Chin & Steven Feiner. Near real-time shadow generation using bsp trees. *Computer Graphics*, 23(3):99–106, July 1989.

[18] Michael S. Paterson & F. Frances Yao. Efficient binary space partitions for hidden-surface removal and solid modeling. *Discrete and Computational Geometry*, 5(5):485–503, May 1990.

[19] Enric TORRES. Optimization of the binary space partition algorithm (bsp) for the visualisation of dynamic scenes. In C. E. Vandoni and D. A. Duce, editors, *Eurographics '90*, pages 507–518, Eurographics Association, 1990.

[20] Bruce F. Naylor. Partitioning tree image representation and generation from 3d geometric models. In *Graphics Interface '92*, pages 201–212, 1992.