Première partie

Représenter et rendre les paysages en synthèse d'images : travaux antérieurs et éléments utiles

CHAPITRE 1

Traiter les scènes naturelles

Les scènes naturelles offrent un domaine d'étude extrêmement vaste et complexe, ne serait-ce par la diversité des formes y prenant place : plantes, terrains, fourrures, nuages, feu, fluides, etc. Un des buts premiers de la synthèse d'images étant de reproduire l'apparence de la réalité, nous disposons ici d'un vaste terrain de jeu où quelques jalons ont été posés, mais où il reste encore beaucoup à faire.

On distingue souvent deux aspects dans la complexité d'une scène naturelle : les objets naturels et les phénomènes naturels. Les objets naturels sont concrets et solides, ils interagissent avec la lumière par des modèles matière-lumière, ce sont par exemple les terrains, les végétaux, etc. Les phénomènes naturels ne sont pas tangibles, leur surface n'est pas clairement définie, ce sont par exemple le brouillard, le feu, le vent, etc. Nous avons choisi de travailler sur les objets naturels d'un paysage, et plus précisément sur les arbres : l'objectif de cette thèse est de traiter la complexité introduite par la présence d'arbres. Cette complexité, qui donne la richesse visuelle d'une image se traduit usuellement par un nombre important de primitives, à l'origine de consommation astronomique de mémoire et de temps de calcul.

Si les objets naturels sont tangibles, les mécanismes entrant en jeu dans leur modélisation et leur apparence sont extrêmement complexes. Par exemple, de nombreux facteurs interagissent entre eux à différentes échelles dans la création d'un arbre, comme la richesse en eau du terrain, la quantité de lumière reçue, etc. En 1, nous passerons en revue les techniques existantes de modélisation de terrains et d'arbres, puis nous verrons qu'il est très vite indispensable de modéliser ces objets à différents niveaux de détails si l'on veut pouvoir les utiliser à grande échelle. En 2 nous détaillerons les techniques classiques pour calculer une image à partir de représentations de ces objets naturels (rendu), ainsi que les techniques de visibilité permettant de diminuer le coût de rendu et de faciliter le calcul des ombres.

1 Modélisation de paysages

Nous nous intéresserons essentiellement aux arbres et aux terrains : nous étudierons les modèles de générations de terrains et d'arbres en 1.1 et 1.2, ainsi que les techniques construisant automatiquement leurs niveaux de détails en 1.3.

1.1 Modélisations et représentations de terrains

La modélisation et la visualisation de terrains ont été beaucoup étudiées, d'autant qu'elles intéressent de nombreuses applications comme la géographie, l'aménagement du territoire, l'exploration spatiale, les jeux vidéos, les simulateurs de vol, etc. Pour la synthèse de paysages, qui nous intéresse au premier plan dans cette thèse, les techniques de génération de terrain sont indispensables puisqu'ils représentent le support de quasiment tous les autres éléments (végétation, ruisseaux, etc.). Deux grandes familles de méthodes de modélisation existent :

- les méthodes extrayant des données réelles issues de mesures des reliefs (Modèle Numérique de Terrain), avec pour problèmes la prise de mesures et le stockage de ces grandes quantités de données ;
- les méthodes générant des reliefs artificiels.

Nous traiterons essentiellement de la génération artificielle de terrains à partir de méthodes procédurales et de leur stockage.

1.1.1 Générations

Les méthodes de génération de terrain sont basées sur la théorie des fractales développée par Manldelbrot [Man78] en 1968, lequel avait remarqué la similitude entre la crête des montagnes et la courbe produite par un mouvement Brownien fractionnaire (*fBm*) à l'origine de sa théorie.

Fournier [FFC82] propose une méthode algorithmique pour le calcul approché du *fBm* basée sur une subdivision récursive du modèle et sur l'introduction d'un facteur aléatoire : pour chaque intervalle non subdivisé, on calcule le point milieu auquel on applique une perturbation aléatoire dont l'amplitude est proportionnelle au niveau de récursivité. Miller [Mil86] proposera une méthode assez similaire en 1986.

Musgrave [MKM89], pour réduire l'apparence uniforme des terrains générés par ces méthodes, apporte deux améliorations :

- l'utilisation des fonctions multi-fractales, qui permettent de prendre en compte les variations hétérogènes d'un même terrain (*e.g.* entre plaines et montagnes) ;
- l'ajout d'un processus d'érosion basé sur un modèle hydraulique : l'eau ruisselle sur le relief en déposant des sédiments ou en érodant la matière. Les temps de calculs sont longs mais les résultats sont très réalistes (*cf.* figure 1.1).

Ces modèles donnent des résultats surprenants de réalisme, mais leur complexité engendre des calculs extrêmement coûteux. Les travaux de Musgrave ont été par la suite, intégré au logiciel *Bryce* [Met] de la société *MetaCreations.*

1.1.2 Représentations

La représentation la plus utilisée pour représenter un terrain est la carte d'élévation[1] (*displacement mapping* [Coo84, MKM89]). Avec cette représentation, un terrain est stocké sous forme

[1] Cette technique est aussi utilisée pour représenter les imperfections d'une surface ou encore les bas reliefs.

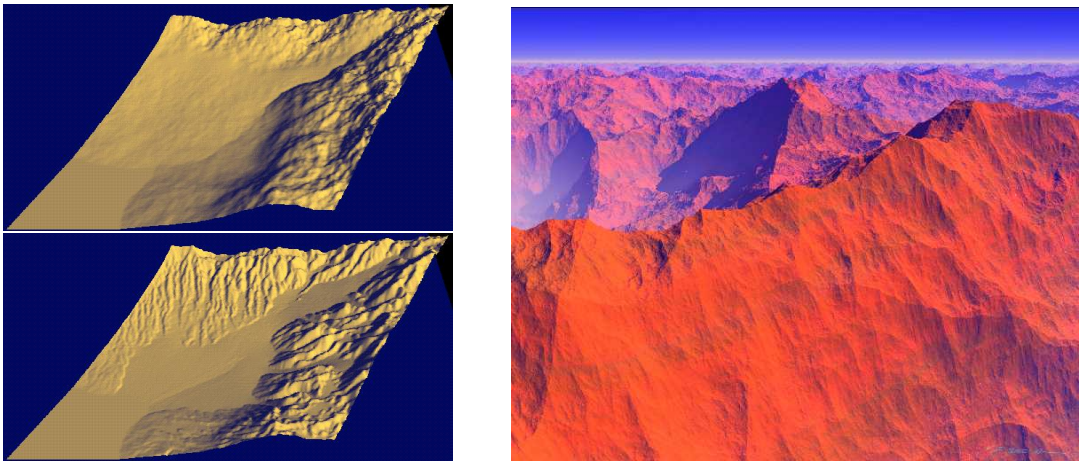

FIG. 1.1 – Le modèle de génération de terrain de Musgrave [MKM89] est basé sur une génération multi-fractales associée à un mécanisme d'érosion. *À gauche* : un exemple d'érosion. *À droite* : un exemple de rendu.

d'un tableau à deux dimensions, où chaque case contient la hauteur du terrain. De plus on peut associer une couleur à chaque case, ce qui revient à associer une texture au terrain. La souplesse de cette représentation provient de la possibilité de considérer ces cartes comme des images 2D dont la manipulation est très intuitive : modélisation possible avec des outils de dessin 2D, facilité de génération de niveaux de détails, etc.

Les travaux présentés ici donnent un bref aperçu des méthodes de modélisation et de représentation des terrains, nous verrons à la section 1.3.1 les techniques de simplification pour construire des niveaux de détails et en 2.1.4 les techniques de rendu et de visibilité qui leur sont spécifiques.

1.2 Modélisations et représentations d'arbres

Après les terrains, les éléments les plus visibles d'un paysage sont les végétaux qui les couvrent, et plus particulièrement les arbres. Il existe aujourd'hui des méthodes capables de construire relativement rapidement, avec plus ou moins de réalisme, la diversité des végétaux d'un paysage. Nous en dénombrons essentiellement trois familles :

- les modèles à base de grammaires ;
- les modèles à base de simulations botaniques ;
- les modèles géométriques.

Ces méthodes construisent dans un premier temps la structure de l'objet, qui est ensuite interprétée pour produire une description géométrique 3D. Nous ne parlerons ici que du processus de modélisation, le rendu étant traité au chapitre 2 (*billboards* 2.3.1, textures volumiques 2.4, système de particules 2.1, images de profondeurs 2.3.4). Nous nous sommes attachés à ces trois familles de modèles parce qu'elles peuvent toutes fournir, plus ou moins facilement, les données de base aux modèles présentés en parties II et III de cette thèse.

Dans cet état de l'art nous ne cherchons pas à être exhaustifs à propos des modèles de génération de végétaux. On peut trouver plus d'informations dans l'état de l'art de la thèse de Chaudy [Cha97], ou dans la catégorisation effectuée par Fournier en 1989 [Fou89].

Les critères auxquels nous porterons attention sont le réalisme du résultat, la diversité des espèces représentables, la nature et la quantité de données nécessaires à la description, les temps de calcul et la prise en compte des influences extérieures (*e.g.* soleil, vent, type de terrain, etc.).

Nous aborderons les modèles à base de grammaires en 1.2.1, suivis des modèles à base de paramètres botaniques en 1.2.2, parce que ce sont les deux familles qui donnent actuellement les meilleurs résultats en terme de diversité et de réalisme. Puis nous verrons un modèle géométrique intéressant car assez simple en 1.2.3 [WP95], qui est un des rares modèles à générer facilement et automatiquement des niveaux de détails.

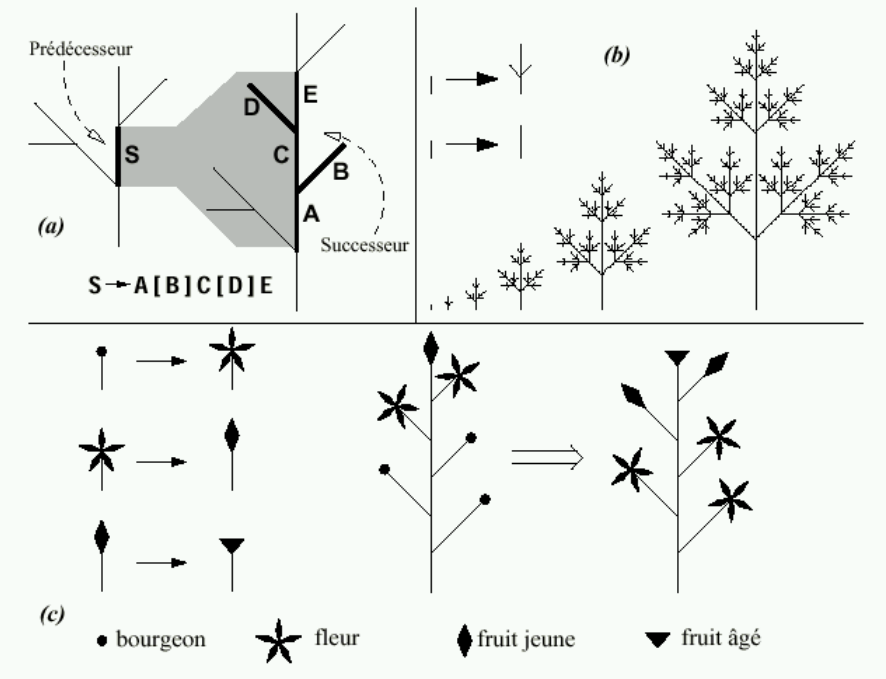

FIG. 1.2 – Principe de réécriture dans les *L-systems* : (a) le motif du prédécesseur est remplacé par le successeur. (b) développement d'une structure ramifiée à partir de deux règles de réécriture. (c) l'évolution des fleurs n'affecte pas la structure sous-jacente. (figure issue de la thèse de Chaudy [Cha97].)

1.2.1 Modèles à base de grammaires

En 1968 Lindenmayer [Lin68] propose un modèle (*L-system*) de développement des structures biologiques basé sur un ensemble de règles de réécriture. Une règle de réécriture est composée de deux membres : le prédécesseur et le successeur. À chaque itération et en parallèle, les occurrences du prédécesseur dans une chaîne sont remplacées par le successeur (*cf.* figure 1.2). Dans le cas des végétaux cette chaîne représente la structure ramifiée de la plante.

Les travaux de Prusinkiewicz [PLH88] ont pris comme base le modèle simple de Lindenmayer et l'ont fortement enrichi. Les processus complexes de croissance de développement sont modélisés grâce à la possibilité des *L-systems* de propager des signaux entre la base et les structures ramifiées. Prusinkiewicz ajoute [PJM94] la prise en compte des facteurs externes comme le voisinage, la recherche de lumière ou l'élagage d'une branche (*cf.* figure 1.3 à gauche). Deussen [DHL$^+$98] utilise le modèle de Prusinkiewicz pour la génération d'un écosystème entier.

Afin de produire plusieurs spécimens différents d'une même espèce de plantes, Prusinkiewicz introduit un facteur aléatoire dans les *L-systems*, ajoutant une probabilité à chaque règle de production (*cf.* figure 1.3 à droite).

La modélisation géométrique des topologies générées se fait en interprétant les symboles de

FIG. 1.3 – *À gauche* : un exemple de croissance soumis à l'élagage. *À droite* : un exemple de scène modélisée par un *L-system*.

la chaîne par un système de type "tortue graphique"[2]. Chaque symbole de la chaîne va, soit modifier la position courante de la tortue, soit appliquer une rotation à la direction courante, soit empiler/dépiler l'état courant du contexte (position, orientation, etc).

Ce modèle de Lindenmayer et Prusinkiewicz pour la génération de plantes s'applique avec succès à de nombreuses espèces. Il est facilement implémentable et extensible, en tout cas dans sa version de base. Cependant, il est difficile à contrôler et nécessite une certaine connaissance morphologique, bien qu'il existe maintenant divers outils faciles d'utilisation comme par exemple *Xfrog* [LD] de la société *Greenworks.*

1.2.2 Modèles à base de paramètres botaniques

De Reyffye [dREF+88] proposent en 1988 un modèle botanique pour la synthèse de végétaux. Cette approche "biologique" de la modélisation des plantes a pour principe la simulation de l'activité des bourgeons sur une échelle de temps discrète. Des probabilités, caractéristiques de l'espèce et du niveau de développement du végétal considéré, sont affectées à chacun des nœuds d'évolutions possibles du bourgeon. Ces données sont issues des connaissances en botanique et en morphologie des végétaux [CIR]. Des facteurs externes peuvent également être intégrés, comme par exemple, la coupe de certaines branches ou des maladies, l'effet de la gravité sur les branches, la présence d'un obstacle entre la lumière et une partie de la plante, etc.

La description issue du moteur de croissance est interprétée comme un ensemble de primitives géométriques simples (troncs de cônes pour les branches et polygones pour les feuilles) qui peuvent ensuite être traitées comme n'importe quel autre objet de synthèse d'images, ce qui facilite leur utilisation par un modeleur pour leur intégration à un paysage, ainsi que par un moteur de rendu.

Ce modèle très complet et réaliste a abouti à un système de simulation développé par le *CIRAD* (Centre de coopération Internationale en Recherche Agronomique pour le Développement) sous l'appellation *AMAP* [CIR], maintenant commercialisé par la société *Bionatics* [Bio].

[2] Par référence au langage *Logo*.

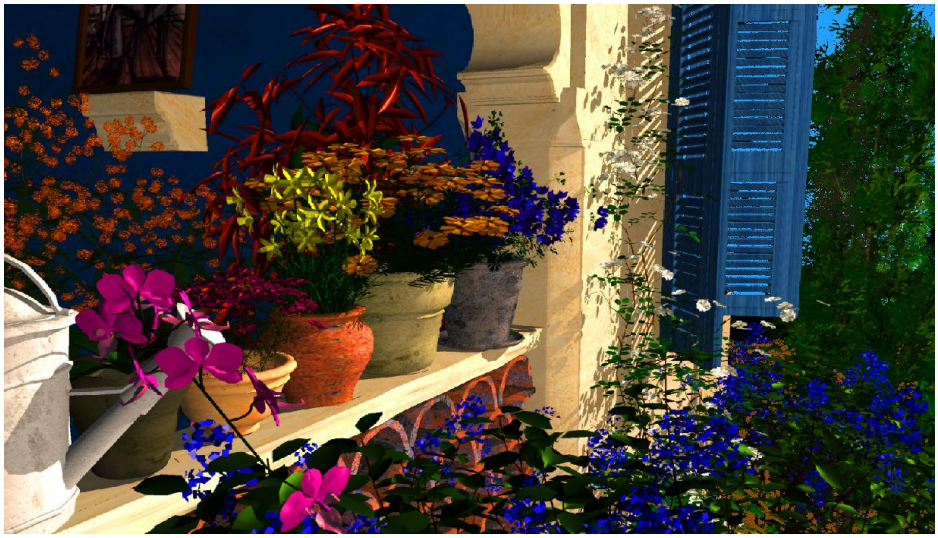

FIG. 1.4 – Exemples de plantes générées par un modèle à base de paramètres botaniques (logiciel *AMAP* issu du CIRAD [CIR], maintenant commercialisé par la société *Bionatics* [Bio]).

1.2.3 Modèles géométriques

En 1995 Weber et Pen [WP95] présentent un modèle intéressant pour la création d'arbres et de leurs niveaux de détails associés. Leur approche est purement géométrique, basée sur un ensemble de paramètres importants qui leur permet d'obtenir des arbres variés et convaincants.

Weber et Pen partent du principe qu'il existe rarement plus de quatre niveaux de récursivité dans la structure ramifiée d'un arbre. Pour chaque niveau de récursivité ils spécifient donc des paramètres géométriques avec une plage de valeurs possibles. Ils prévoient aussi des paramètres pour prendre en compte l'influence du vent, du soleil ou de la taille des branches.

L'aspect intéressant de leur modèle est la possibilité de génération d'une géométrie à complexité variable, ce qui permet d'utiliser leur arbre dans un moteur de rendu à niveaux de détails (*cf.* figure 1.5). Nous reparlerons de niveaux de détails dans le cas des arbres à la section 1.3.1.

1.2.4 Bilan global

Ces modèles de génération de plantes offrent des résultats réalistes grâce à la prise en compte de nombreux facteurs. Les inconvénients propres à l'ensemble de ces modèles sont la nécessité de connaître les paramètres botaniques et morphologiques des plantes (un logiciel comme *AMAP* est fourni avec une large bibliothèque réaliste constituée par le *CIRAD*), ainsi que la lourdeur des descriptions 3D produites qui oblige à utiliser des niveaux de détails pour optimiser la phase de rendu.

1.3 Niveaux de détails

Nous venons de voir les outils de construction importants pour la synthèse de paysages : la génération de terrain et la génération de végétaux. Ces méthodes donnent de bons résultats en terme de réalisme, par contre il est évident que la complexité en nombre de primitives devient vite astronomique si l'on veut représenter tout un paysage : plusieurs centaines de milliers de polygones sont nécessaires pour un arbre, et donc plusieurs milliards pour un paysage entier. Ceci

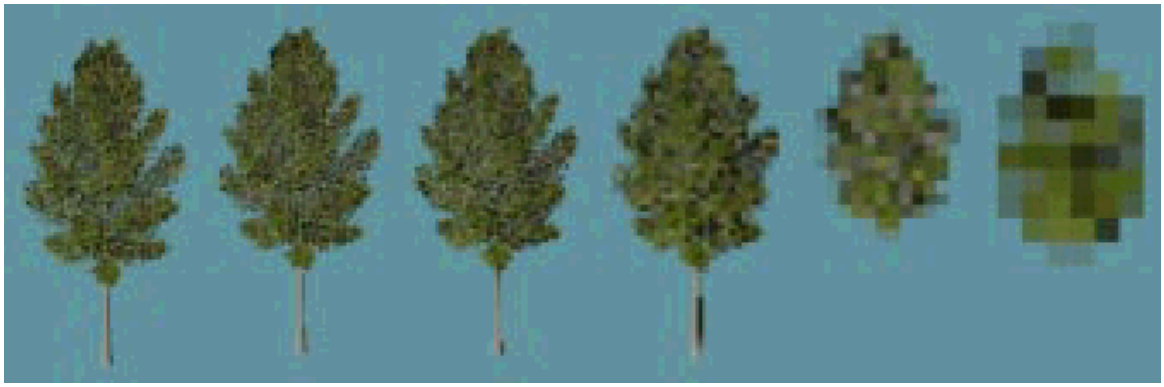

FIG. 1.5 – Exemple de niveaux de détails générés par le modèle géométrique de Weber et Pen [WP95].

augmente énormément le coût de rendu, voire le rend irréalisable. De plus, lorsque des milliers de petites primitives contribuent à la couleur d'un même pixel, les traiter toutes est extrêmement coûteux. Bien souvent, le moteur de rendu se contente de les échantillonner (*cf.* section 2) mais alors, se pose le problème de l'aliassage (*cf.* Étude de cas, section 3.2).

Une solution, devenue standard, est l'utilisation de niveaux de détails : on utilise plusieurs représentations d'un objet, de moins en moins complexes en nombre de primitives au fur et à mesure que l'objet s'éloigne. L'approche classique est de remplacer N polygones suffisamment équivalents par un seul, moins coûteux. Dans le principe, cette notion de niveaux de détails est très efficace. Dans la pratique, la construction des différents niveaux pose de nombreux problèmes : il faut que la transition entre deux niveaux soit continue, qu'aucun artefact ne soit visible à l'œil.

Nous verrons en 1.3.1, que pour les objets polygonaux, il existe de nombreuses techniques de simplifications plus ou moins adaptées à différentes familles d'objets. Nous nous attacherons en particulier au cas des terrains et des arbres, puis nous finirons en 1.3.2 par les techniques de transitions existantes entre différents modèles.

1.3.1 Objets polygonaux à plusieurs niveaux

Pour diminuer le coût de rendu ainsi que l'aliassage, une méthode consiste à entretenir plusieurs descriptions géométriques plus ou moins précises de chaque objet, afin d'utiliser la représentation la plus adaptée à l'échelle. Une variante plus élaborée se charge d'adapter dynamiquement le maillage. Un premier problème réside dans la construction automatique des représentations allégées, notamment par des méthodes de décimation de polygones. Un second problème concerne la continuité du rendu lorsqu'on s'éloigne ou qu'on se rapproche de l'objet.

Nous verrons que des techniques existent dans le cas des objets "pleins" (*i.e.* aux surfaces fermées) mais que dans le cas des arbres, d'autres solutions doivent être trouvées.

Cas général

De nombreux travaux portent sur la simplification de maillage d'objets polygonaux ([Hop96, Hop97], etc.). Initialement, leur domaine d'application était la simplification de modèles 3D scannés, qui comportent souvent des millions de polygones (largement redondants). Il devient alors important d'alléger ces représentations pour les rendre utilisables. Ces techniques ont dérivé vers la fabrication automatique de niveaux de détails, souvent avec succès pour certaines classes d'objets (*cf.* figure 1.6).

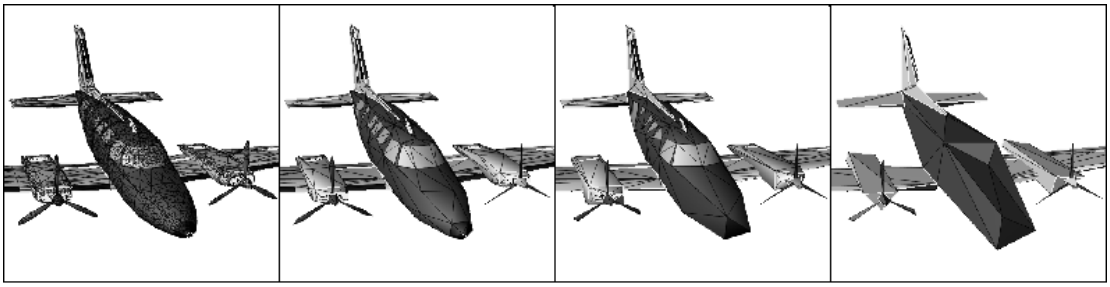

FIG. 1.6 – La simplification de maillage donne de bons résultats avec des objets “pleins”, pour preuve cet avion simplifié par la technique de Hoppe [Hop96].

Cas des terrains

Nous avons vu en 1.1.1 que les terrains sont représentés par des cartes d'élévation pour des facilités de modélisation et de stockage. Pour le rendu, il est nécessaire de convertir la carte d'élévation en série de polygones. La grande quantité de polygones ainsi générés impose de considérer des techniques de simplification pour obtenir un maillage adapté au relief du terrain. En effet, sans simplification, une carte de résolution 256×256 sera représentée par un maillage de $256 \times 256 \times 2 = 131072$ triangles alors que tous les terrains ne nécessitent pas partout une telle précision. Par exemple, une grande zone plate constituée de nombreux triangles pourra avantageusement être remplacée par un polygone unique. Des travaux allant dans ce sens ont été réalisés par Garland [GH95]. Souvent, une deuxième phase de simplification [LKR+96, DWS+97, Hop98] est nécessaire juste avant le rendu pour tenir compte de la position de l'observateur et ainsi satisfaire aux critères de temps réel. De plus, comme la simplification est dynamique, ces modèles garantissent une continuité entre les niveaux.

Cas des arbres

Dans le cas d'un arbre polygonal, il s'agit de remplacer les nombreux polygones représentant des feuilles par quelques uns. Dans le modèle de Weber et Pen, dont nous avons parlé à la section 1.2.3, les auteurs construisent la hiérarchie de niveaux de détails comme ceci : dans l'ordre descendant des niveaux de détails (*i.e.* lorsque l'on s'éloigne progressivement), les branches sont progressivement interprétées comme des lignes et les feuilles comme des points, puis toujours en s'éloignant, certaines branches ou feuilles ne sont plus interprétées (*cf.* figure 1.5). Cela revient à supprimer les branches les moins significatives au fur et à mesure que l'on s'éloigne. Ceci permet une génération automatique et efficace ; cependant, la suppression des feuilles et des branches au fur et à mesure que l'on descend dans la hiérarchie n'est pas très réaliste : il est évident que l'illumination et l'opacité générale de l'arbre va s'en trouver complètement changée et la continuité entre les niveaux ne pourra être garantie.

Bilan

Ces techniques de construction de niveaux de détails par simplification de maillage donnent des résultats efficaces et visuellement continus dans le cas d'objets “pleins” ou dans le cas de terrains, par contre, la nature même de ces techniques comporte une certaine limitation. Par exemple, une tôle ondulée vue de loin sera simplifiée en un plan, ce qui, du point de vue géométrique, peut sembler raisonnable, ne l'est plus du tout du point de vue de la lumière : la manière dont cet objet

réagit à la lumière ne dépend pas de la distance. L'effet visuel moyen des deux modèles sera totalement différent, ce qui se fera sentir lors de la transition par un phénomène de saut dans l'apparence (*poping*). De même une zone dense en feuilles remplacée par son enveloppe géométrique verra sa transparence et son illumination changer. La contribution de nombreuses feuilles ombrées visibles au cœur de la zone disparaîtra.

1.3.2 Transitions entre représentations

Comme nous le disions en introduction un problème essentiel dans l'utilisation des niveaux de détails est la continuité lors de la commutation des modèles. Une solution classique pour forcer cette continuité est de mélanger de manière pondérée les images résultantes de deux niveaux lors de la transition. Cette technique a l'avantage d'être simple et de donner des résultats continus avec toutes les représentations, mais elle est deux fois plus coûteuse puisqu'il faut tracer les deux niveaux à la fois lors de la transition et engendre souvent des artefacts (*e.g.* "écho" de certains détails qui se sont légèrement déplacés entre deux niveaux).

Kajiya dans [Kaj85] introduit la notion de *hiérarchie de modèles*, consistant à passer d'une représentation géométrique à une représentation texturelle, puis photométrique au fur et à mesure que l'on s'éloigne. Nous aborderons durant le chapitre 2 sur les représentations dédiées plusieurs modèles qui se prêtent particulièrement bien à ces transitions : par exemple le modèle de transition entre différentes représentations de détails de Becker (*cf.* section 1.1.8), le modèle des textures volumiques (*cf.* section 2.4), etc. A noter le modèle de Perbet [PC01] destiné à la représentation et à l'animation de prairies en utilisant trois niveaux de détails correspondant à trois représentations : géométrie, représentation 2D $1/2$ et texture. Les résultats sont continus, même lorsque l'herbe se couche sous le vent.

2 Rendu

Nous venons de passer en revu les techniques de générations de terrains et de plantes, ainsi que les techniques associées générant des niveaux de détails, en vue d'alléger les coûts de rendu souvent énormes pour les scènes d'extérieur, et de diminuer les artefacts visuels (l'aliassage) allant de pair avec cette complexité.

Nous allons décrire maintenant les techniques de rendu permettant de passer de l'espace objet 3D à l'espace image 2D, ainsi que les techniques connexes : calcul des ombres, problèmes de visibilité [3] et optimisations.

Nous verrons donc dans cette section consacrée au rendu les techniques classiques de rendu en 2.1 suivies par un aperçu des méthodes de visibilité en 2.2.

2.1 Techniques de rendu

On peut découper le travail de rendu en deux phases : déterminer les surfaces visibles à l'écran, puis calculer la couleur de l'élément de surface (ou la couleur moyenne des éléments) apparaissant en chaque pixel en tenant compte des caractéristiques d'orientation et de matière de celles-ci, ainsi que des conditions d'éclairage. De nombreuses techniques ont été développées dans ce but ; nous ne détaillerons que les deux les plus utilisées, à savoir le tampon de profondeur (*Z-buffer*) en 2.1.1

[3] La détection des faces cachées depuis l'observateur, ainsi que la détermination des ombres, sont deux problèmes de visibilité traités au moment du rendu.

et le lancer de rayons (*ray-tracing*) en 2.1.2. Nous parlerons du rendu volumique en 2.1.3 et nous finirons par les techniques de rendu spécifiques au terrain en 2.1.4.

2.1.1 Le Z-buffer et ses extensions

L'origine du *Z-buffer* est donnée à Catmull [Cat74]. Le principe consiste à laisser à une extension du concept de mémoire d'image[4] le soin de déterminer la visibilité des surfaces. Le *Z-buffer* est une mémoire, "un tampon des profondeurs" qui sert à stocker la profondeur de chaque pixel visible à l'écran. Durant le rendu, la profondeur d'un élément candidat à paraître dans le pixel (fragment) est comparée à la valeur de la profondeur déjà stockée pour ce pixel : si cette comparaison indique que le nouveau fragment est devant celui stocké, alors celui-ci remplace le contenu du pixel écrit dans la mémoire image et le *Z-buffer* est mis à jour. La complexité de cet algorithme est proportionnelle aux nombres de primitives traitées et à leur surface à l'écran ; pour en améliorer encore l'efficacité, Greene propose une version hiérarchique dans [GK93].

De nombreuses extensions à cet algorithme existent, la plus intéressante est peut être le *A-buffer* [Car84] qui conserve pour chaque pixel une liste des fragments candidats à y apparaître et calcule pour chaque fragment la proportion du pixel qu'il recouvre. Ceci permet de traiter les problèmes d'aliassage et d'intégrer des objets semi-transparents.

Une autre amélioration [HA90] est le tampon d'accumulation (*accumulation buffer*) qui permet d'accumuler plusieurs images en déplaçant légèrement la caméra entre chaque rendu, soit pour réduire l'aliassage, soit pour obtenir un effet de flou de mouvement (*motion blur*) ou de mise au point (profondeur de champ).

L'autre solution pour réduire l'aliassage est de subdiviser les pixels. La couleur finale du pixel est obtenue en moyennant les sous-pixels. À noter que cette technique revient à effectuer le rendu à une résolution supérieure de celle de l'image (sur-échantillonnage).

Cet algorithme ne traite pas les ombres, ni les réflexions et le traitement de la transparence nécessite un tri des polygones avant de les envoyer à la carte graphique. Un fragment de transparence α et de couleur C_f ayant réussi le test de profondeur sera composé de la manière suivante avec la couleur C_p déjà stockée dans le pixel : $C_p = \alpha \times C_f + (1 - \alpha) \times C_p$, ce qui correspond bien à un calcul de transparence si tous les fragments se trouvant derrière celui-ci on déjà été tracés (d'où le tri des fragments en pré-traitement).

Un des avantages du *Z-buffer* est la possibilité de réaliser facilement des implémentations matérielles très efficaces. Aujourd'hui les moteurs graphiques à base de *Z-buffer* permettent le rendu et l'affichage de scènes de plusieurs centaines de milliers de polygones en temps réel. Diverses techniques ont été développées pour obtenir des ombres à l'aide de cartes d'ombre (*cf.* section 2.2.2), ou à l'aide de volumes d'ombre [Cro77, Ber86, Sla92, McC00].

2.1.2 Lancer de rayons

Le principe du lancer de rayons (*ray-tracing*) [Whi80, GAC+89] est de calculer les objets visibles en remontant le chemin de la lumière parvenant à la caméra, *i.e.* en lançant des rayons à travers tous les pixels de l'image (fonctionnement inverse d'un appareil photo). Comme le principe du lancer de rayons est très simple à programmer, on peut lui faire simuler toute l'optique géométrique et ainsi prendre en compte de nombreuses caractéristiques optiques comme la réflexion, la réfraction, l'ombrage, etc. Notamment les reflets de la scène sur une surface lisse sont obtenus en envoyant un rayon secondaire depuis la surface dans la direction miroir à celle d'arrivée par

[4] Une mémoire d'image sert à stocker les attributs (couleur) de chaque pixel de l'espace image.

rapport à la normale. Comme le *Z-buffer*, la deuxième partie du processus consiste alors à calculer la couleur que vont avoir les objets en faisant appel à un modèle d'illumination (*cf.* chapitre 2 section 1.1) pondéré par l'éclairage (*i.e.* l'ombrage). Le principe de calcul des ombres est identique : pour savoir si un objet est éclairé, on lance un rayon d'ombre vers les sources de lumière.

Le coût de cet algorithme est important : pour chaque pixel de l'écran il est necessaire de lancer un rayon, voir plusieurs avec la technique de sur-échantillonage (*cf.* section 2.1.2). Avec un lancer de rayons non optimisé, pour chaque rayon on calcule son intersection avec toutes les primitives de la scène. De plus, pour chaque primitive intersectée d'autres rayons sont lancés (réflexions, ombres, etc.). Pour diminuer le nombre d'intersections à calculer, et donc le coût, il est indispensable de recourir à des techniques d'optimisations (*cf.* section 2.1.2).

Un des problèmes important que doit traiter le moteur de rendu est l'aliassage. Un arbre à lui seul compte plusieurs centaines de branches et plusieurs milliers de feuilles ; on peut donc imaginer la quantité de primitives que compte une forêt et donc le nombre moyen de primitives par pixel (le pire étant à l'horizon). Lors du rendu d'une image il est évident que toutes les primitives ne sont pas visibles explicitement, par contre elles contribuent toutes à l'aspect global de la scène (par exemple les aiguilles d'une forêt de sapins ne sont pas toutes visibles individuellement mais si on les supprime il ne reste que les branches !). Plusieurs centaines de primitives contribuent à la couleur d'un pixel, il faut donc, à défaut de toutes les traiter, en considérer suffisamment pour diminuer au maximum les problèmes d'aliassage.

Sur-échantillonage

La solution couramment utilisée à ce problème d'aliassage en lancer de rayons est le sur-échantillonage. Cela revient à lancer plusieurs rayons par pixel, distribués aléatoirement, puis à moyenner les résultats pour donner la couleur finale. Le nombre de rayons lancés peut-être adaptatif, c'est-à-dire varier d'un pixel à l'autre en fonction du nombre d'objets présents dans le pixel : ce nombre n'étant pas connu à l'avance, on décide, après avoir lancé plusieurs rayons, de poursuivre tant que l'écart type des couleurs est supérieur à un certain seuil paramétrable. Cette méthode statistique donne un bon rapport efficacité/coût pour des scènes d'intérieur ou des scènes peu fouillées, c'est-à-dire lorsque le nombre de rayons lancés par pixel n'excède pas dix ou vingt. Cependant, dès que la complexité de la scène est trop importante son efficacité est limitée : soit on lance un nombre de rayons suffisant, mais les temps de calcul explosent, soit l'aliassage se fera rapidement sentir, surtout lors du calcul d'animation (car la cohérence temporelle n'est pas respectée).

Lancer de faisceaux

L'autre solution au problème d'aliassage est de calculer l'intégrale de l'ensemble des objets se trouvant dans le pixel : c'est ce que se propose de faire le lancer de faisceaux [HH84] (*beam-tracing*) ou de cônes [Ama84] (*cone-tracing*) en lançant un rayon pyramidal ou conique par pixel. À noter que le lancer de rayons avec sur-échantillonnage, que nous avons vu au paragraphe précédent, est une approximation numérique de cette intégrale.

L'avantage de cette technique de *beam-tracing* est de capturer tous les objets d'un pixel en lançant un unique rayon, l'inconvénient est qu'elle augmente la complexité du calcul pour chaque primitive. En effet, l'intersection entre une pyramide (ou un cône) et une primitive de la scène est plus compliquée à calculer que l'intersection entre une droite et cette primitive, comme c'est le cas avec le lancer de rayons classique. De plus, pour un rayon représenté par une droite, le calcul

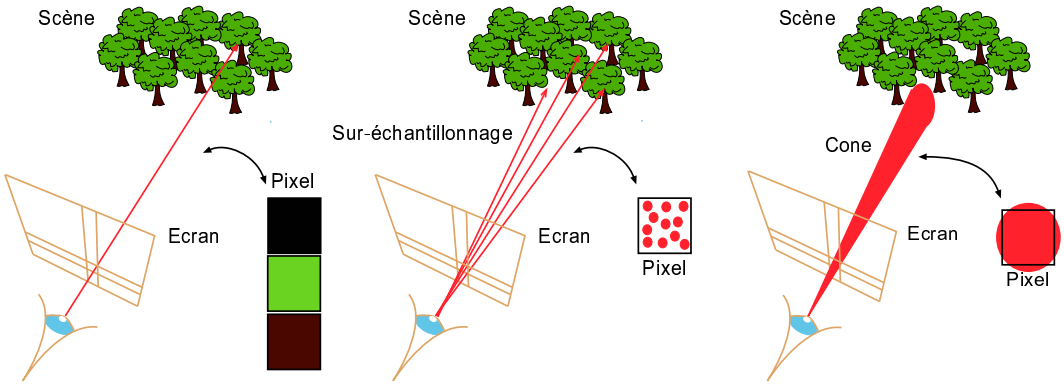

FIG. 1.7 – De nombreuses primitives se projettent dans un pixel. *À droite* : en ne lançant qu'un rayon par pixel la couleur déterminée sera aléatoire (aliassage). *Au milieu* : la solution est d'échantillonner suffisamment le pixel en lançant de nombreux rayons. *À droite* : on lance un unique rayon conique pour considérer tous les objets qu'il contient.

de l'illumination s'arrête à l'intersection de l'objet le plus proche de l'œil, alors que dans le cas du *cone-tracing*, on doit tenir compte de la surface du pixel recouverte par la primitive, et continuer avec les autres primitives se trouvant derrière s'il reste une portion non couverte.

Le sur-échantillonnage est plus simple à implémenter que le *beam-tracing*, mais pour une scène comportant de nombreuses primitives, petites par rapport à la taille d'un pixel une fois projetées, la technique de *beam-tracing* donnera de meilleurs résultats en terme d'aliassage et sûrement même en terme de coût de calcul.

Optimisations

Les techniques d'optimisation du lancer de rayons sont nombreuses. Le principe général est la subdivision de l'espace objet avec pour but la limitation du nombre d'intersections rayon-objet à calculer. Les deux principales techniques sont les volumes englobant (sphères ou boîtes) et les grilles, éventuellement récursives (octree) [SB87, JW89]. De nombreux travaux portent sur ce domaine, pour un éventail de toutes ces techniques, se référer à [GAC+89, Hai].

2.1.3 Rendu volumique par lancer de rayons

Le rendu volumique est destiné à calculer une image 2D d'une structure de données représentée par une grille 3D (chaque case étant appelée un voxel). Un exemple typique de données est une grille où chaque voxel contient une densité (*e.g.* une densité de tissus humains).

La technique classique de rendu est basée sur le principe du lancer de rayons que nous avons vu précédemment. Le rayon parcourt le volume de données en utilisant un algorithme de tracer de droite discrète et accumule au fur et à mesure la transparence, ainsi que la couleur si celle-ci est une donnée de la grille. Pour obtenir la quantité de lumière que reçoit chaque voxel il faut lancer un rayon vers chaque source de lumière (comme dans le cas du lancer de rayons classique).

Nous verrons à la section 2.5.1 du chapitre 2 qu'il existe une technique de rendu volumique plus efficace utilisant un rendu par tranches.

2.1.4 Cas des terrains

Nous avons vu à la section 1.1.2 qu'une représentation commode d'un terrain est la carte d'élévation (*displacement map*), mais de nombreux moteurs de rendu préfèrent traiter des polygones, il

faut alors convertir la carte en géométrie au moment du rendu.

D'autre part, il existe des techniques dites d'*inverse displacement* [PHL91, Tai92], sorte de lancer de rayons adapté, gérant directement l'intersection entre le rayon et la représentation, sans expliciter la carte d'élévation en facettes. Cela permet en outre de le faire efficacement, en procédant de manière hiérarchique : le principe consiste à organiser les données en quadtree et à pré-calculer dans chaque case l'altitude maximale. On a alors la garantie que le rayon ne coupe pas cette portion de terrain si ses points d'entrée et de sortie dans la case sont plus hauts que ce maximum, sinon, on repose le problème sur les quatre cases filles, et ainsi de suite. Quand on atteint la résolution des données, il faut tester l'intersection du rayon et de la surface dans la case en la calculant vraiment, ce qui n'arrive que pour peu de cases.

Une autre technique [PH96], non spécifique au lancer de rayons, basée sur un système de cache, permet de ne pas expliciter toute la géométrie : les polygones sont générés au vol, on ne convertit que les données utiles dans la zone en cours de traitement. On utilise un système de cache pour éviter de générer des polygones à plusieurs reprises.

FIG. 1.8 – *À gauche* : deux images de Lindstrom [LKR+96] illustrant un maillage fin et un maillage grossier que l'on peut utiliser comme niveaux de détails. *À droite* : une image de Premoze [PTS99] illustrant la possibilité d'éclairer un terrain montagneux à différents moments de l'année.

Pour en finir avec le rendu de terrains, nous citons un article de Premoze [PTS99] permettant de synthétiser un paysage de montagne à différents moments de l'année et de la journée. La texture initiale du terrain est extraite de photos. Puis, la méthode permet de simuler la tombée et la fonte de neige ainsi que les différentes illuminations que peut prendre le terrain en fonction de sa composition et de l'éclairage (*cf.* figure 1.8 à droite).

2.2 Visibilité

La visibilité tient une part importante dans l'algorithmique de la synthèse d'images. Ses applications sont, entre autres, la suppression des parties cachées pour la détermination de l'ombrage (optimisation).

Nous avons vu dans les paragraphes traitant de la modélisation qu'un paysage peut compter des milliards de polygones, ce qui est bien trop pour un moteur de rendu actuel, même câblé par le matériel d'une carte graphique. D'autre part, la majorité des ces primitives n'est pas visible au final car dans le dos de l'observateur, derrière une montagne, derrière un arbre, cachés en arrière plan, etc. Seulement, les supprimer automatiquement est un problème difficile pour un ordinateur. La

visibilité offre donc des algorithmes permettant de n'envoyer au moteur de rendu que les primitives visibles.

Nous proposerons au chapitre 6 une structure pré-calculant l'information de visibilité pour le calcul des ombres. En 2.2.1, nous parlerons donc des techniques de visibilité en général, nous verrons en 2.2.2 la technique des *shadow map* permettant le calcul rapide de la visibilité pour les ombres, puis nous finirons en 2.2.3 avec des techniques de visibilité spécifiques aux terrains.

2.2.1 Généralité

Le but ici n'est pas d'être exhaustif à propos des travaux sur la visibilité, domaine très actif en ce moment. Un état de l'art complet peut être trouvé dans [Dur99].

Historiquement, la suppression des objets cachés est une des grandes motivations des algorithmes de visibilité (*cf.* Jones [Jon71] ou Clark [Cla76]). Les techniques les plus simples sont la suppression des objets se trouvant en dehors du champ de la caméra (*frustum culling*) qui permet de supprimer jusqu'à 75%, voir 80% des primitives, et la suppresion des parties d'un objet se trouvant à l'arrière (*backface culling*). Les techniques plus complexes capables de supprimer les objets cachés par d'autres (*occlusion culling*) sont souvent basées sur un pré-calcul de structures de visibilité (*e.g.* cartes), et permettent de supprimer jusqu'à 95% des objets avec un coût consacré aux tests bien moindre (ceci dépend bien sûr de la configuration de la scène et de la position de l'observateur). Airey [ARB90] et Teller [TS91] furent les premiers à proposer de pré-calculer la visibilité dans des environnements architecturaux intérieurs. Ils exploitent le fait que si l'on est dans une pièce, on ne voit les autres que par les portes ou fenêtres, *i.e.* en construisant une structure de vues (portails) qui prend en considération les zones visibles. Ces techniques pour ce type de scènes sont très efficaces, mais grandes consommatrices de mémoire. Des algorithmes capables de traiter des scènes d'extérieur ont fait leur apparition ces dernières années [COFHZ98], où la notion d'occludeurs est utilisée pour éliminer les zones cachées à la vue. Ces algorithmes sont basés sur la notion de cellules de visibilité : l'espace est découpé en octree et, pour chaque cellule (voxel), on pré-calcule les parties visibles de la scène. Ces techniques fonctionnent correctement lorsque les occludeurs sont de grandes tailles, ce qui n'est pas le cas des arbres où les occludeurs (*i.e.* feuilles) sont petits et nombreux. Durand [DDTP00] et Schaufler [SDDS00] introduisent la notion de fusion de plans d'occlusion, qui permet de grouper les zones contiguës cachées par chaque occludeur (*e.g.* feuille) en des zones plus grandes.

2.2.2 Cartes d'ombres

Nous venons de passer en revue les techniques de pré-calcul de la visibilité dans le but de ne traiter que les objets visibles. Le calcul des ombres est un problème assez similaire, puisqu'il revient à déterminer les objets visibles depuis les sources de lumière, avec pour avantage que celles-ci sont souvent fixes dans le temps.

La technique des cartes d'ombres (*shadow map*), introduite par Reeves en 1987 [RSC87], pré-calcule la visibilité depuis la source de lumière dans une carte, en effectuant un rendu de la scène depuis cette source et en conservant le tampon de profondeur. Au moment du rendu d'un objet on calcule la distance entre l'élément vu dans le pixel et la source de lumière. Si cette distance est supérieure à celle stockée dans la *shadow map* alors l'objet est à l'ombre, sinon il est à la lumière.

Lorsque la source de lumière change de position ou lorsqu'un objet bouge, il faut recalculer la carte, ce qui oblige à effectuer deux rendus : un depuis la source de lumière, et un depuis l'œil. Grâce aux performances des cartes graphiques actuelles le temps réel est possible, d'autant que

la plupart des effets peuvent être désactivés pour la construction de la *shadow map* (illumination, texture, etc.) et les niveaux de détails grossiers utilisés. À noter que cette technique ne fonctionne qu'avec des objets opaques, car la carte ne donne que l'objet le plus proche de la source. Un autre inconvénient de cette technique est que la résolution de la texture, ainsi que la dynamique limitée des profondeurs stockées, restreint la précision des ombres. Seules certaines cartes graphiques de haut niveau (Onyx^2 Infinite Reality) disposent de cette fonctionnalité.

Une technique similaire à celle-ci est l'*alpha shadow map* qui n'utilise que des cartes de transparence, et s'applique quand il y a une séparation entre les objets projetant leur ombre (les occludeurs) et les objets ombrés. Lors du calcul de la carte, seuls les objets occludeurs sont rendus. Le tampon est ensuite utilisé comme texture multiplicative sur les objets ombrés (*i.e.* la couleur est pondérée par l'opacité stockée dans la carte). Contrairement à la technique des *shadow maps*, les *alpha shadow maps* autorisent des occludeurs semi-opaques, ne demandent pas de fonctionnalité spécifique du matériel graphique et ne posent pas le problème de la dynamique des profondeurs. Par contre, elles demandent de bien séparer (et donc de connaître) les occludeurs des objets ombrés (ce qui est par exemple le cas d'arbres projetant leur ombre sur un terrain).

Des techniques ont été développées pour combiner les aspects positifs des *shadow maps* et des *alpha shadow maps*. C'est-à-dire être capable de gérer des objets transparents ou une densité d'objets très importante par rapport à la résolution de la texture, sans avoir à séparer les occludeurs des objets ombrés. Lokovic [LV00] calcule l'ombrage et l'auto-ombrage d'une chevelure, dont les primitives sont à une résolution trop fine pour la méthode de base. Il découpe l'espace en intervalles de profondeur auxquels il associe une carte. Il appelle l'ensemble *cartes d'ombres en profondeur* (*deep shadow map*). Son implémentation est complètement logicielle et n'est donc plus temps réel. Tae-Yong [KN01] en accélère le calcul en utilisant le matériel graphique, sans pour autant parvenir à atteindre le temps réel. Soler [SS98] propose une technique pour calculer une *alpha shadow map* contenant des ombres douces. Il convolue à l'aide du matériel graphique des images des occludeurs prises depuis différentes positions de la source de lumière.

2.2.3 Cas des terrains

Nous présentons ici des méthodes traitant plus spécifiquement de visibilité de scènes de terrains dans le but d'en calculer l'auto-ombrage. Ces techniques sont également utilisées pour limiter le nombre de polygones envoyés au moteur de rendu.

Max introduit la notion de carte d'horizon (*horizon map*) dans [Max88] pour ajouter de l'ombrage à une surface dont les normales sont représentées par une carte de perturbations (*bump-map*[5]). L'idée est de calculer la visibilité du ciel à chacun des points échantillonnés de la surface. Max considère huit directions autour de chacun de ces points et détermine l'horizon en utilisant les points échantillonnés (*cf.* figure 1.9 à gauche). Durant le rendu, un point est considéré comme illuminé si le soleil est visible de ce point : pour cela, il utilise la direction échantillonnée la plus proche. La pénombre est calculée à partir de la quantité du disque solaire vu depuis le point. La limitation de cette méthode est le sous-échantillonnage des directions de visibilité qui entraîne des artefacts.

Cabral dans [CMS87] utilise des cartes d'horizon similaires pour calculer la *Bidirectional Reflectance Distribution Function BRDF* (*cf.* section 1.1.6) d'une carte de perturbation des normales (*bump-map*). Il utilise 24 directions de vue et projettent les triangles du terrain à la place des points

[5] Le *bump-mapping* est la perturbation des normales d'une surface sans changer la géométrie (*cf.* chapitre 2 section 1.1.2). Le plus souvent on utilise une carte de normales comme texture de la surface.

lors du calcul de la visibilité, ce qui diminue le sous-échantillonnage mais ne le fait pas disparaître entièrement.

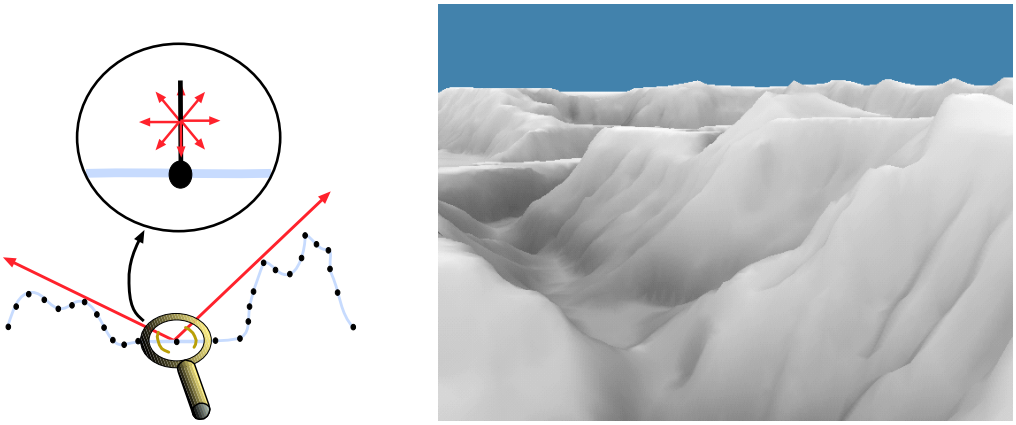

FIG. 1.9 – Cartes d'horizon. *À gauche* : pour chaque point échantillonné de la surface, Max [Max88] calcule dans 8 directions la hauteur de l'horizon. *À droite* : Stewart [Ste98] augmente la précision des cartes d'horizon et les utilise pour calculer l'ombrage et accélérer le rendu en n'envoyant à la carte graphique que les parties visibles du terrain.

Cohen-Or [COS95] utilise une technique d'échantillonnage similaire pour découper le terrain en partie visible et partie invisible dans le but de limiter le nombre de primitives envoyées au moteur de rendu. Stewart dans [Ste97, Ste98] augmente la précision de la méthode en considérant tous les points de la surface pour le calcul de la visibilité (*cf.* figure 1.9 à droite), alors que Max ne considérait que huit directions. En utilisant une hiérarchie il introduit un algorithme rapide lui permettant de calculer les cartes d'horizon rapidement, même pour un très grand nombre d'échantillons.

3 Bilan

Les techniques de modélisation de terrains et d'arbres donnent des résultats remarquables en terme de réalisme, même s'il reste encore des travaux à effectuer, notamment pour les vues rapprochées (écorces, terre, poussière, sable, etc.). Cependant, les modèles générés sont extrêmement lourds de polygones. Les techniques de rendu sont bien incapables de traiter convenablement (*i.e.* sans coût de calcul prohibitif et sans aliassage) les milliards de primitives qu'engendrent les arbres.

Les algorithmes de simplification de maillage sont capables de générer efficacement des niveaux de détails pour les terrains, mais leur application aux arbres n'est pas convaincante, notamment à cause de la nature fouillée et dispersée des feuilles. Même si les techniques de visibilité diminuent le nombre de polygones envoyés au moteur de rendu et pré-calculent l'information pour l'affichage des ombres, il semble difficile de traiter rapidement des paysages composés d'à peine quelques dizaines d'arbres.

Il est évident, au regard de ces techniques, qu'il y a avant tout un problème de représentation, au-delà de la simple hiérarchisation qui est déjà poussée à bout dans toutes les techniques actuelles.

CHAPITRE 2

Traiter la complexité

Au cours du chapitre précédent, nous avons traité des représentations dédiées à la synthèse de paysages (*i.e.* terrains et végétaux). Le principal problème qui découle de celles-ci est le traitement des milliards de primitives générées, que le rendu doit traiter. La construction de niveaux de détails par simplification de maillage améliore la situation pour les terrains, par contre, leur utilisation est impossible pour les végétaux du fait de leur structure dispersée. Cette lacune doit être comblée par d'autres chemins, *i.e.* il nous faut trouver d'autres méthodes capables de représenter la complexité de manière plus efficace que les polygones.

L'objectif de ce chapitre est donc de présenter diverses représentations existantes traitant la complexité en synthèse d'images sous toutes sortes de formes (rugosités, fourrures, forêts, nuages, etc). Ce panel de techniques aux approches originales et variées (formes analytiques, octrees, points, textures 3D, etc.) offre des solutions à de nombreux problèmes ; nous verrons cependant qu'il n'existe pas une solution unique traitant tous les problèmes. Bien qu'elles ne soient pas toutes en rapport direct avec la synthèse de paysages, ces techniques présentent un concept ou une philosophie utile à la compréhension des "contributions" que nous présenterons durant cette thèse.

Le plan de ce chapitre est donné par l'échelle des phénomènes à représenter : nous traiterons à la section 1 des complexités microscopiques, puis nous détaillerons à la section 2 les modèles dits *alternatifs*, traitant de complexité à échelle macroscopique.

1 Complexité microscopique

La quête du réalisme fait que l'on cherche à enrichir l'apparence visuelle des surfaces, par exemple en essayant de représenter la rugosité. Le polygone est la primitive de base en synthèse d'images, mais représenter une surface granuleuse par des milliards de micro-polygones n'est pas viable : on a donc cherché à représenter cette complexité à l'aide de représentations mieux adaptées. De nombreux travaux fournissent des formules analytiques ou des tables donnant l'illumina-

tion d'une surface, chacun tenant compte d'une caractéristique nouvelle par rapport au précédent [FvDFH90].

La même philosophie de conception est aussi utilisée pour l'illumination des volumes, par exemple, pour représenter les milieux participatifs comme les nuages, ou les milieux denses en géométrie comme les poils d'une fourrure. Dès que la géométrie est trop complexe au point qu'il ne soit plus raisonnable de la représenter explicitement ou dès que le phénomène n'a pas de surface définie, il devient nécessaire de concevoir des représentations plus efficaces, souvent basées sur une intégration analytique. La conception de ces représentations est le résultat d'une modélisation mathématique poussée, souvent inspirée de techniques que la physique a déjà mises au point et qui fournissent un bon point de départ. Ces méthodes nous intéressent, car cette philosophie de représenter la complexité par une intégration analytique sera aussi la notre, dans la première contribution de cette thèse (*cf.* partie II).

Nous traiterons en 1.1 des modèles d'illumination de surfaces, puis nous décrirons en 1.2 les modèles d'illuminations volumiques.

1.1 Modèles d'illumination d'une surface

Les modèles d'illumination de surface (*shaders*) sont nombreux ; nous ne nous intéresserons qu'à quelques-uns d'entre eux, plus directement reliés aux représentations que nous présenterons dans les contributions de cette thèse. Nous verrons :

- en 1.1.1 le modèle empirique mais classique de Phong qui est souvent le modèle de départ lorsque l'on cherche à intégrer l'illumination totale d'un objet complexe ;
- en 1.1.3 le modèle de Torrance et Sparrow, qui fut un des premiers à intégrer analytiquement les micro-facettes d'une surface ;
- en 1.1.4 divers modèles d'illumination analytiques ;
- en 1.1.5 le modèle anisotropique de Poulin, qui se base sur l'intégration analytique de l'illumination d'une surface couverte de micro-cylindres, problématique proche de ce que nous allons présenter par la suite ;
- en 1.1.6 la fonction de distribution de la réflectance bidirectionnelle (*bidirectionnal réflectance distribution function BRDF*) ;
- en 1.1.7 la notion de fonction bidirectionnelle de texture qui est utilisée pour traiter l'illumination de surfaces à partir de mesures réelles ;
- en 1.3.2 le modèle de Max permettant la transition douce entre différents niveaux de *bump*.

1.1.1 Modèle d'illumination de Phong

Le modèle d'illumination de surface le plus connu, car le plus utilisé en synthèse d'images, est certainement celui de Phong [FvDFH90]. Ce modèle empirique et simple décrit le calcul de la couleur d'un objet en fonction de sa couleur intrinsèque et des sources lumineuses. Plus précisément, il ajoute au modèle diffus de Lambert une contribution spéculaire représentant les reflets de la lampe calculés en fonction de la normale, et des directions de vue et de lumière :

$$\begin{aligned} I &= \mathrm{ambient} + \mathrm{diffus}_{\mathrm{Lambert}} + \mathrm{speculaire}_{\mathrm{Phong}} \\ I &= I_a k_a + I_p (k_d (N.L) \mathbb{1}_{N.L>0} + k_s (R.V) \mathbb{1}^n_{R.V>0}) \end{aligned}$$

avec N la normale à la surface, L la direction de la lumière, V la direction de vue, R le vecteur symétrique de V par rapport à la normale N (*cf.* figure 2.1 à gauche) et n le coefficient de spécularité,

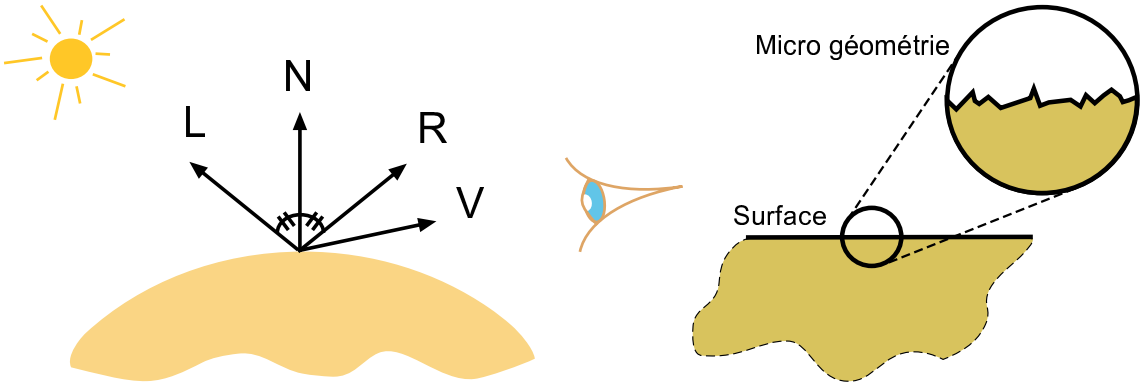

FIG. 2.1 – *À gauche* : les données pour le modèle d'illumination empirique de Phong [FvDFH90]. *À droite* : la micro-géométrie d'une surface.

inversement proportionnel à la rugosité.

Blinn a apporté une formulation alternative de l'illumination de Phong qui utilise le vecteur moyen H du vecteur de vue V et du vecteur de lumière L. (H est aussi appelé le vecteur de direction maximum de spécularité). Le terme de réflexion spéculaire, utilisé en pratique, s'écrit alors $(N.H)^n \mathbb{1}_{N.H>0}$.

Ce modèle a l'avantage d'être simple, facile à implémenter, supporté de nos jours par toutes les cartes graphiques 3D[1], et de donner des résultats visuellement acceptables même si inexacts, ce qui explique son succès.

1.1.2 Cartes et textures de perturbations

Une façon d'augmenter la complexité visuelle tout en minimisant la complexité des traitements est d'utiliser une texture contenant l'image des détails [Cat74, BN76]. Une texture est un tableau 2D que l'on plaque (*mapping*) sur une surface. Dans la version la plus simple, ce tableau contient des couleurs (voir de la transparence) pour former, soit une image unique qui vient s'appliquer sur toute la surface, soit un motif que l'on va répéter pour couvrir celle-ci.

Blinn représente la complexité d'une surface sans la modéliser par des polygones, en introduisant la notion de texture de paramètres de Phong [Bli77] qui permet par exemple, de simuler différents matériaux ou les variations de rugosité d'une plaque de métal tout en utilisant un unique polygone. D'autre part, il introduit l'idée de texture de perturbation de normales (*bump-mapping*) [Bli78], qui revient à perturber les normales pour simuler l'apparence d'une surface non plane sans augmenter le nombre de polygones (*cf.* figure 2.2). Comme aucun relief n'est réellement créé avec cette technique, il n'y a pas d'ombres portées, et la silhouette de l'objet reste lisse.

Un polygone texturé peut se voir soumis à l'aliassage si la résolution de la texture se trouve être plus fine que celle de l'écran. Dans ce cas l'aliassage est plus simple à traiter que pour les polygones (*cf.* chapitre 1 section 2.1.2), car on connaît le voisinage. Williams [Wil83] introduit la technique du *mipmapping*, il pré-calcule la texture à diverses résolutions et, lors du rendu, utilise celle qui est adaptée à la distance, *i.e.* dont le pixel de texture est le plus proche du pixel de l'écran.

[1] Ce modèle est évalué aux sommets des polygones par les cartes graphiques classiques (*i.e.* ne disposant pas de la fonctionnalité d'illumination par pixel). Les valeurs des couleurs en chaque pixel sont alors interpolées lors du remplissage du polygone.

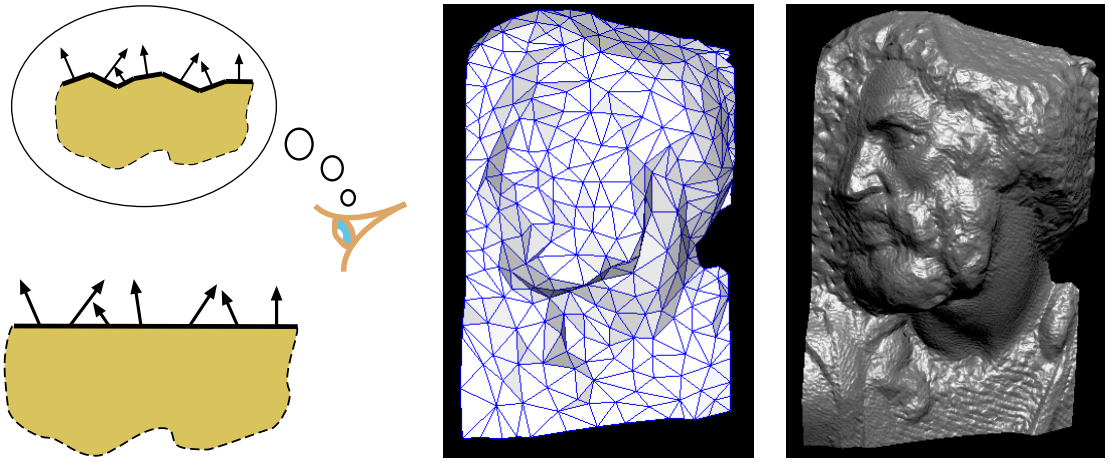

FIG. 2.2 – Carte de perturbations de normales (*Bump-mapping*). *À gauche* : seules les normales à la surface sont perturbées pour donner l'illusion d'un relief authentique. *Au milieu* : le maillage nu. *À droite* : le maillage enrichi par la technique de *bump-mapping*, donnant l'illusion d'un relief fin.

1.1.3 Intégration analytique des micro-facettes d'une surface

Rapidement, des modèles plus complexes et plus proches de la réalité (au sens physique du terme) ont été développés : un de ces premiers modèles, issu de la physique, fut celui de Torrance et Sparrow [TS66, TS67], introduit par Blinn et Cook en synthèse d'images dans [CT82, Bli77]. C'est un modèle analytique basé sur l'idée qu'une surface isotrope est constituée de micro-facettes parfaitement réfléchissantes et distribuées aléatoirement. Torrance et Sparrow estiment statistiquement la réflexion de la lumière dans une direction donnée. Ce type de surface est qualifié d'isotrope car l'intensité de la lumière réfléchie est indépendante de l'orientation de la surface par rapport à la normale, ce qui ne sera pas le cas des situations que nous allons envisager maintenant.

1.1.4 Divers modèles d'illuminations

De nombreux autres modèles d'illumination analytiques existent. Notamment, le modèle anisotrope de Kajiya [Kaj85] basé sur une nouvelle intégration des formules de Beckmann de diffusion d'une surface rugueuse ; le modèle de Lewis [Lew94] qui essaie de rendre les modèles d'illumination classiquement utilisés en synthèse d'images physiquement réalistes en les comparant à des critères ou des données réelles (*BRDF*) ; le modèle de Schlick [Sch94a] dont la paramétrisation forte permet de représenter toutes sortes de caractéristiques (isotrope, anisotrope, homogène, etc.) ; une généralisation anisotrope du modèle de Lambert par Oren qui représente une surface par de nombreuses surfaces lambertiennes [ON94], etc. On trouvera une synthèse de ceci dans l'état de l'art de Schlick [Sch94b].

1.1.5 Modèle d'illumination anisotrope de Poulin

Poulin dans [PF90] propose un modèle analytique anisotrope de réflexion et réfraction basé sur l'utilisation de micro-cylindres. L'anisotropie est représentée par de petits cylindres distribués sur la surface, pour simuler des rayures ou des fils collés. Différents niveaux d'anisotropie peuvent ainsi être obtenus en faisant varier la distance entre deux cylindres ou en plaçant plus ou moins haut la surface par rapport aux axes (*cf.* figure 2.3).

Cette fonction de réflectance anisotrope est évaluée dans [PF90] par le calcul analytique de l'intégrale du modèle d'illumination de Phong sur les cylindres et le plan. Cette approche analytique est calculée de manière exacte pour le terme diffus et avec une approximation pour le terme spéculaire.

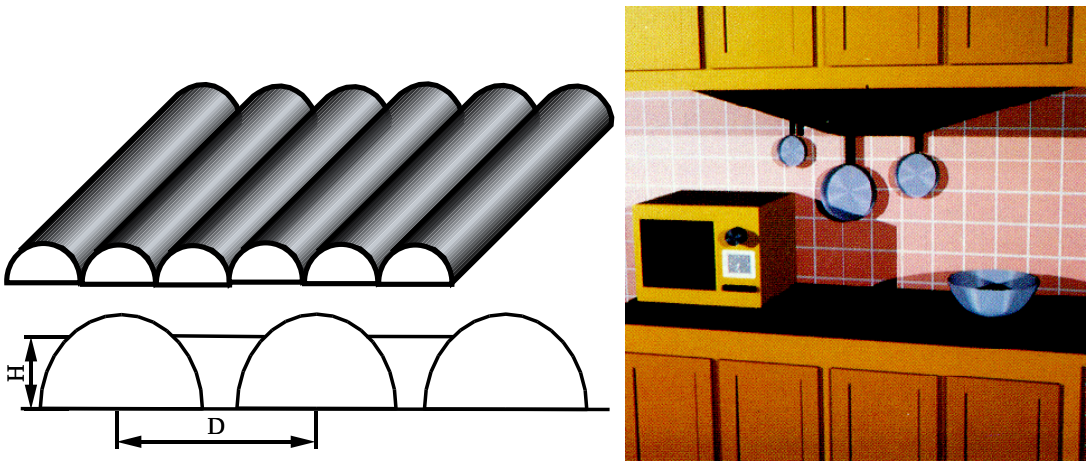

FIG. 2.3 – Modèle d'illumination anisotrope de Poulin [PF90]. *À gauche* : le degré d'anisotropie est contrôlé par H et D. *À droite* : un exemple d'illumination anisotrope (fonds des casseroles).

1.1.6 Fonction de distribution de la réflectance bidirectionnelle (BRDF)

Pour reproduire la complexité d'une surface visible par sa réflectivité, une autre approche consiste à recourir à des mesures réelles du comportement optique d'un matériau, obtenues au moyen d'un gonioréflectomètre. Il y a potentiellement une valeur de réflexion différente pour chaque direction incidente de lumière et chaque direction d'observation, chacune repérée par deux angles. On obtient ainsi une table à quatre dimensions représentant la fonction de distribution de la réflectance bidirectionnelle (*bidirectionnal reflectance distribution function BRDF*). De nombreux travaux portent sur ce domaine, voir [Rus97] pour plus de détails.

De nombreux modèles d'illumination sont basés sur cette technique des *BRDF*, comme le modèle de Gondek [GMN94] dépendant de la longueur d'onde, le modèle de Westin [WAT92] qui approchent une *BRDF* par une méthode simulant la diffusion (*i.e.* Monte Carlo) sur la géométrie, etc.

1.1.7 Fonction bidirectionnelle de texture (BTF)

Dana dans [DvGNK99] propose d'utiliser une base de données [DvGNK] d'images de surfaces réelles photographiées sous toutes les conditions, associée à des techniques de traitement d'images pour en reconstruire la fonction de réflectance. La fonction bidirectionnelle de texture (*Bidirectionnal Texture Function BTF*) qu'introduit Dana associe à un couple (direction de vue, direction de lumière) une image. Cette fonction, qui est formellement 6D, peut être construite en échantillonnant n directions de vue et m directions de lumière (*cf.* figure 2.4).

Lors de notre deuxième contribution (*cf.* partie III) nous réutiliserons l'idée de prendre des images d'un objet depuis plusieurs points de vue, avec pour chacun de ces points de vue différentes positions de lumière.

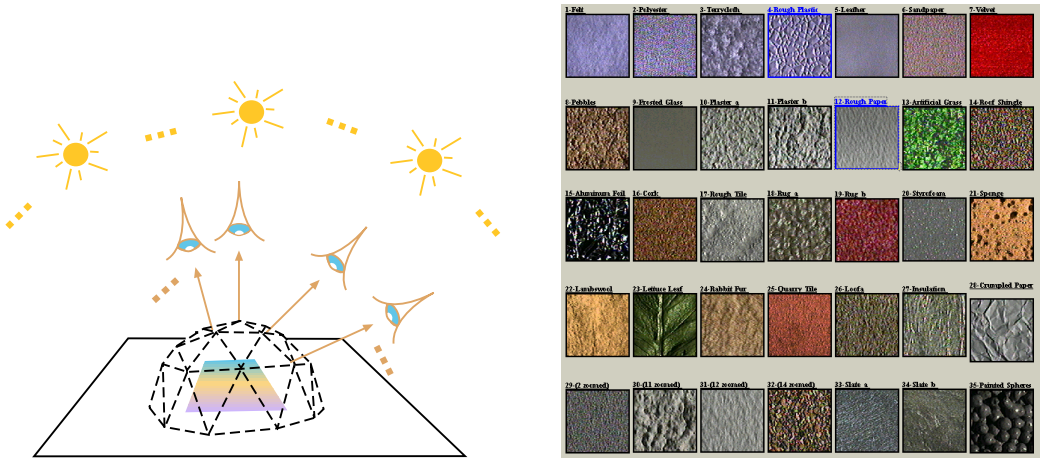

FIG. 2.4 – Fonction bidirectionnelle de texture (*BTF*) [DvGNK99]. *À gauche* : la surface est prise en photo depuis plusieurs points de vue et avec plusieurs directions de lumière. *À droite* : un exemple de surfaces réelles que propose la base de donnée des Universités de Columbia et Utrecht [DvGNK].

1.1.8 Transitions continues entre représentations des détails

En complément de la section 1.3.2 du chapitre 1, qui traitait de la transition entre les représentations, nous abordons ici le cas des transitions entre modèles de *bump*, proposé par Becker et Max dans [BM93].

Cabral et Max [CMS87] utilisent les cartes d'horizon (*cf.* chapitre 1 section 2.2.3) pour apporter l'information de visibilité manquante aux cartes de *bump-mapping* et ainsi tenir compte des interactions du voisinage, même pour des surfaces enrichies par le *bump mapping*.

Les techniques de *BRDF* et de cartes de déplacement (*displacement map*) permettent de tenir compte de l'auto-ombrage, puisque celui-ci est une propriété intrinsèque de ces modèles. Dans [BM93] Becker et Max utilisent ces deux représentations, plus celle du *bump mapping* enrichie de l'auto-ombrage que nous venons de voir, comme différents niveaux de détails. Ils proposent une méthode de transition efficace en tenant compte de la distance à l'œil, de l'angle de vue et de la fréquence de la *bump-map* pour choisir la bonne représentation.

Plus récemment Heidrich [HS99] propose une solution pour implémenter ces techniques de *bump* en temps réel, en utilisant les cartes graphiques standards avec un rendu multi-passes. Toujours en s'aidant du matériel graphique, il apporte une solution temps réel à l'absence d'auto-ombrage du *bump-mapping* [HDKS00].

1.2 Modèles d'illumination volumique

Simuler l'illumination de gaz (nuages, brouillard, etc.) en représentant explicitement les millions de particules qui le composent n'est pas viable, même si une discrétisation grossière peut être utilisée à grande échelle (*e.g.* grille volumique). En revanche, le calcul d'un modèle intégrant analytiquement cette complexité à petite échelle sera efficace. Il en va de même pour de nombreux autres phénomènes complexes comme le ciel, la fourrure, etc.

Le principe d'intégration analytique de l'illumination d'un volume est similaire au cas d'une surface : on intègre un modèle d'illumination simple au sens mathématique du terme sur toute la géométrie du volume en tenant compte de la visibilité et de l'auto-ombrage (qui sont encore plus importants pour un volume que pour une surface).

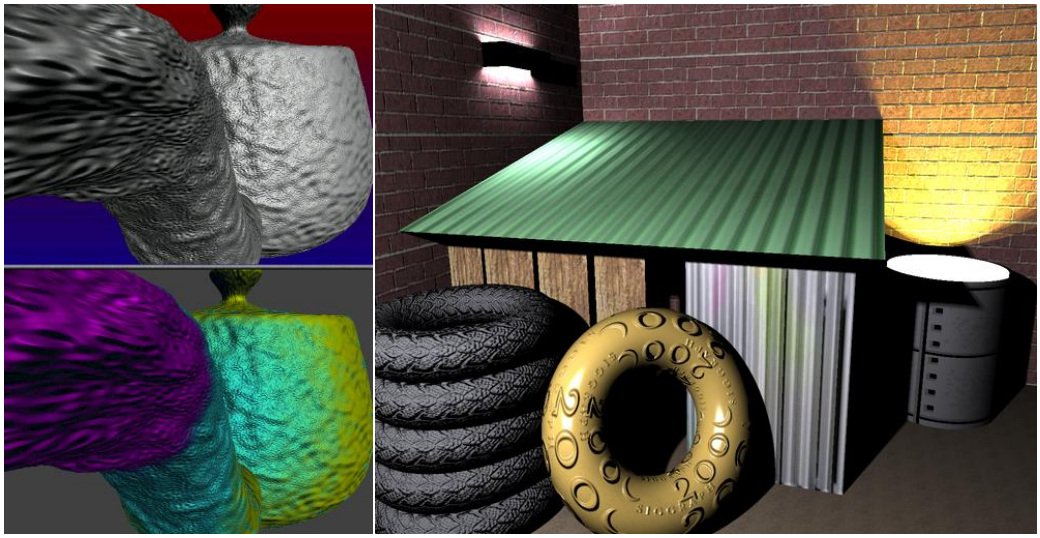

FIG. 2.5 – *À gauche* : transition douce entre les trois modèles : *displacement mapping* en rouge (vue de près), *bump-mapping* en bleu, et *BRDF* en jaune (vue de loin) [BM93]. *À droite* : Heidrich [HDKS00] ajoute les ombres et l'illumination indirecte au *bump-mapping* en temps réel.

Nous présentons en 1.2.1 les modèles d'illumination analytiques basés sur une distribution de particules, suivis en 1.2.2 par un modèle analytique de fourrure.

1.2.1 Modèles d'illumination basés sur une distribution de particules

Blinn dans [Bli82] fut l'un des premiers à proposer un modèle analytique d'illumination en milieu participatif. Son modèle est basé sur une répartition aléatoire dans l'espace de micro-sphères qu'il appelle particules. Il dispose de plusieurs paramètres comme l'épaisseur de la couche de particules, le nombre de sphères par unité de volume, le rayon d'une sphère, qui lui permettent de calculer la probabilité d'intersection entre un rayon et une particule, ainsi que la transparence globale de la couche. La brillance (albédo) et la fonction de phase définissent la transmission de la lumière par une particule en fonction des directions d'observation et d'éclairage. Il calcule analytiquement la quantité de lumière émise dans une direction donnée en ajoutant à la lumière diffusée par les particules, la lumière traversant le milieu (*cf.* figure 2.6).

Les modèles analytiques cherchant à représenter les gaz sont peu nombreux, outre [Bli82] que nous venons de voir, on citera [Sta94] où Stam propose un algorithme de rendu stochastique de gaz (nuages, fumée et feu) représenté par un champ de densité aléatoire. De nombreux autres travaux existent concernant la représentation de nuages mais ce sont pour la plupart des représentations basées sur la simulation. Pour plus d'informations sur les techniques de représentations de nuages, se reporter à [Ebe01, Pro].

Dans le même esprit que le modèle d'illumination analytique de Blinn [Bli82], on citera un modèle de Stam [Sta01] dédié à la représentation de la peau, en prenant en compte les diffusions multiples dans une tranche bornée par deux surfaces rugueuses. Il calcule d'abord une solution discrète aux équations de transfert radiatif, puis par mise en correspondance des courbes (splines) il construit le modèle d'illumination analytique qu'il applique au rendu de peau (*cf.* figure 2.6).

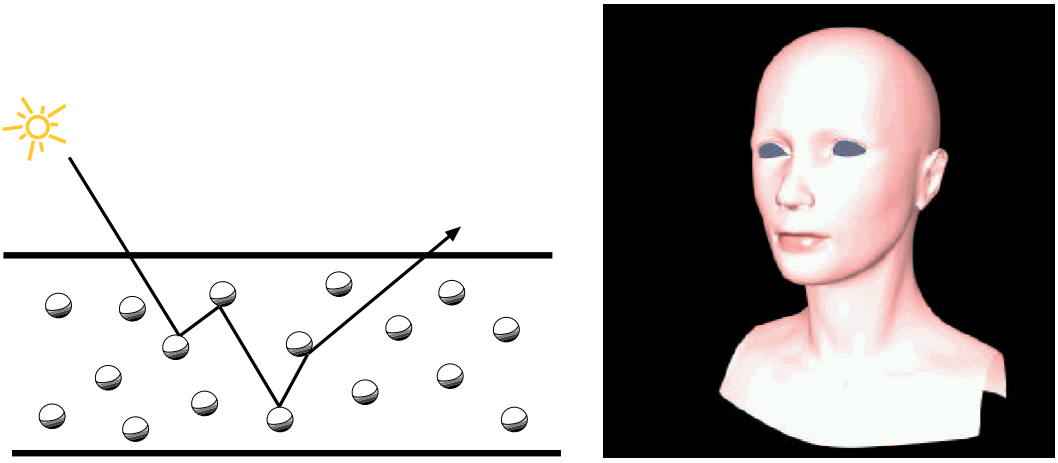

FIG. 2.6 – *À gauche* : Blinn [Bli82] modélise les nuages comme un espace rempli de particules aléatoirement distribuées puis en calcule analytiquement un modèle d'illumination. *À droite* : Stam [Sta01] construit un modèle analytique d'illumination de peau en mettant en correspondance les résultats discrets des équations de transferts radiatifs avec des courbes analytiques.

1.2.2 Modèle de fourrure : illumination d'un cylindre

Miller [Mil88] pré-calcule, dans une table, l'illumination d'un cylindre (à la manière d'une *BRDF*), mis à part qu'il profite de la symétrie axiale pour limiter la quantité de données à stocker. Son but est d'afficher des objets formés d'armatures arrondies semblables à des portions de cylindres. Il applique aussi cette méthode à la fourrure, les poils étant représentés par des cylindres (*cf.* figure 2.7).

Une année après les travaux de Miller, Kajiya et Kay [KK89] intègrent analytiquement l'illumination d'un cylindre, nous détaillerons leur modèle à la section 2.4. Poulin [PF90] avec son modèle d'illumination de surfaces (*cf.* section 1.1.5) utilise aussi l'intégration analytique de l´illumination d'un cylindre pour développer un modèle d'illumination anisotrope.

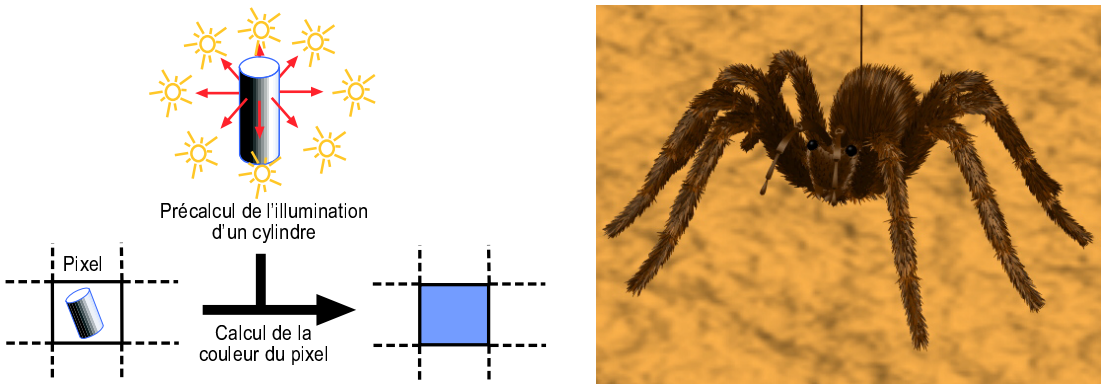

FIG. 2.7 – Illumination d'un cylindre [Mil88]. *À gauche* : Miller pré-calcule, dans une table, l'illumination d'un cylindre à la manière d'une *BRDF*, mis à part qu'il utilise la symétrie axiale d'un cylindre pour stoker un minimum de données. Plus tard Kajiya et Kay [KK89] puis Poulin [PF90] calculerons cette illumination analytiquement. *À droite* : "Borris l'araignée poilue" réalisé à l'aide du modèle de d'illumination d'un cylindre de Miller.

1.3 Bilan des modèles d'illumination

Nous avons vu différents modèles d'illumination surfaciques, puis différents modèles volumiques, lesquels traitent d'une certaine forme de complexité. Le point marquant de cet aperçu est que ces modèles utilisent très souvent une intégration de la complexité d'où découle une formule analytique. L'avantage de ce type d'approche est de fournir une formule prête à être utilisée, facilitant l'implémentation du modèle.

Bien sûr, toute complexité n'est pas toujours facilement intégrable, au sens mathématique du terme : intégrer requière de modéliser la répartition et l'effet des composantes (*e.g.* les particules d'un gaz) sous forme mathématique, ce qui suppose de disposer d'un cadre à priori, de faire des hypothèses, de se ramener éventuellement à une représentation statistique, pour ensuite trouver une forme analytique (éventuellement approchée) de l'intégrale. Le calcul formel, souvent complexe, doit se résoudre sans trop d'approximations, en faisant attention de ne pas faire disparaître ainsi les caractéristiques essentielles de ce que l'on cherche à modéliser. À cause de ces difficultés de modélisation et de réalisation du calcul formel nécessaire à l'élaboration du modèle analytique, on se contente souvent des techniques de discrétisation, plus faciles à mettre en place, mais souvent plus coûteuses en temps de calcul.

Les techniques que nous venons de voir conduisent à une formule ou à une structure de données simple correspondant à l'illumination d'un phénomène ou d'une géométrie. Cependant, il manque l'autre partie nécessaire à la représentation de l'apparence, *i.e.* les formes macroscopiques des objets. Si nous prenons l'exemple des gaz, nous disposons de modèles analytiques d'illumination assez efficaces, mais il nous manque les techniques de modélisation et de stockage de la forme du nuage. Nous allons voir maintenant des représentations que nous qualifions d'alternatives (aux représentations polygonales), qui combinent ces deux aspects pour donner une représentation complète.

2 Représentations alternatives

Les méthodes que nous avons décrites dans les paragraphes précédents avaient toutes pour but d'intégrer les effets de la complexité locale d'une surface ou d'un volume, souvent sous une forme analytique. Mais l'illumination ne suffit pas à représenter le phénomène à grande échelle, il faut l'associer à une forme, à laquelle l'illumination va s'appliquer : pour une surface le modèle d'illumination peut être associé à un polygone, mais pour représenter des phénomènes plus complexes (*e.g.* fourrure, arbres), il faut être capable d'en décrire l'apparence.

On représente les objets par des polygones car leur rendu est simple avec un lancer de rayons ou un *Z-buffer*. Les polygones sont mal adaptés aux objets volumiques (nuages) ou quasi-volumiques, tant la densité de détails est grande (fourrure, feuillage, herbe sur une prairie, etc.). Ils existent une série de représentations plus ou moins spécialisées, que nous qualifierons d'alternatives, qui se proposent de traiter la complexité de manière différente sans recourir nécessairement aux polygones.

Ces modèles se basent sur des représentations très variées, et ils ne traitent pas toujours de paysages. Cependant, ils offrent des solutions élégantes à la prise en charge efficace de la complexité. Cette famille de modèles a été une forte source d'inspiration pour la conception des techniques présentées dans la partie "contributions" de cette thèse ; par conséquent, il est important que nous les détaillions ici.

Nous verrons dans un premier temps l'utilisation de deux primitives très simples, le trait et le point en 2.1 et en 2.2 puis, nous survolerons les techniques utilisant l'image comme support

(*Image Based Rendering IBR*) en 2.3. Nous détaillerons le principe des textures volumiques en 2.4, ainsi que les techniques à base de tranches en 2.5, et nous finirons en 2.6 par une technique alternative représentant de la macro-géométrie par des textures (*e.g.* osier).

2.1 Système de particules

Les systèmes de particules [Ree83, RB85, Cha97], furent parmi les premiers modèles à ne pas utiliser les polygones comme primitives de base : cette technique est adaptée aux géométries complexes dont les détails fins peuvent être représentés par des trajectoires, comme l'eau vive, le feu ou les végétaux.

FIG. 2.8 – Système de particules [Ree83, RB85]. *À gauche* : l'illumination et l'auto-ombrage se calculent à partir des distances Da et Dd. *À droite* : un exemple de paysage généré par cette technique.

L'idée première [Ree83] est de représenter des géométries complexes ou difficilement représentables avec des primitives géométriques classiques, par une primitive très simple mais en très grands nombres : le trait. Les particules évoluent dans l'espace en subissant différentes influences (gravité, lois de croissance, etc.) et forment ainsi des trajectoires. Il n'y a pas d'interaction entre les particules dans le modèle initial.

Reeves et Blau présentent en 1985 [RB85] des applications des systèmes de particules pour la modélisation et le rendu d'éléments naturels : arbres, prairies, etc. L'une des nouveautés est que les particules peuvent interagir entre elles (*e.g.* collisions). Un des aspects intéressant de cet article est la manière dont Reeves et Blau calculent l'illumination et l'auto-ombrage d'un arbre. Une représentation probabiliste de l'arbre est associée à la structure, et est utilisée pour estimer l'ensoleillement d'une particule en fonction de sa position dans l'arbre et de la position du soleil : plus la particule se trouve à l'intérieur de l'arbre, plus elle est à l'ombre (*cf.* figure 2.8 à gauche). Cette approche empirique donne un rendu localement approximatif mais, à cause du très grand nombre de particules l'impression globale est excellente (*cf.* figure 2.8 à droite).

Outre la spécificité de son champ d'application, une des limites de cette technique est son incapacité à modéliser une structure pour une espèce d'arbre donnée. En outre, la spécificité de son algorithme de rendu, qui dessine les particules comme des traits de crayon, diffère totalement de la représentation classique et rend assez difficile son intégration à une scène existante.

2.2 Rendu à base de points

Bien que l'idée d'utiliser le point comme primitive de rendu soit apparue dès 1985 [LW85], ce n'est que très récemment que ce domaine a pris de l'ampleur. En 2000 Pfister [PZvBG00] et Rusinkiewicz [RL00] la revisitent en montrant que le "disque", très facilement géré par une carte graphique, est une primitive bien adaptée à la représentation de nombreux phénomènes ou d'objets complexes.

À la place des polygones, Pfister utilise des points qu'il appelle des *surfels*, compression de *surface elements*. Ces points ne sont pas connexes, ce qui permet d'en adapter le nombre en fonction de la taille de l'objet à l'écran. Le coût d'affichage d'un objet est proportionnel à sa taille à l'écran et non à sa complexité intrinsèque. Ce qui permet de conserver un taux de rafraîchissement de l'image contant, même avec beaucoup d'objets affichés.

Tandis que Pfister convertit les objets polygonaux en *surfels* en pré-traitement, Stamminger [SD01] échantillonne à la volée des objets procéduraux, ce qui offre une plus grande souplesse dans le choix de la précision. Il montre aussi que cette technique est bien adaptée au rendu de phénomènes naturels variés comme une montagne, le mouvement de l'eau ou le rendu d'arbres.

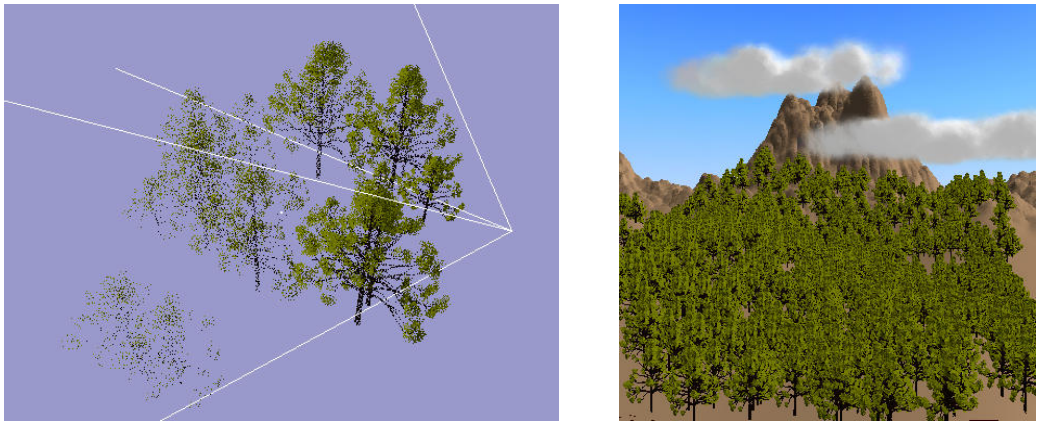

FIG. 2.9 – *À gauche* : des arbres représentés par des points. Plus on s'éloigne de la caméra moins il y a de points affichés. *À droite* : un exemple de paysage rendu avec la technique des points [SD01].

Cette technique est jeune et comporte certains défauts : texturation limitée, objets semi-transparents difficilement représentables, le calcul par moyenne des normales lorsqu'on simplifie le modèle est contestable, etc. Néanmoins l'idée semble très prometteuse, pour preuve la quantité de travaux en cours sur ce sujet [MZG01, PG01].

2.3 Modèles à base d'images

Toujours dans l'idée que certains phénomènes sont trop complexes pour être représentés explicitement en détails (*i.e.* avec beaucoup de polygones) nous survolerons les techniques de rendu à base d'images (*Image Base Rendering IBR*). L'idée est d'utiliser la complexité potentielle que cristallise une image pour représenter un phénomène. La deuxième technique que nous présenterons en partie III de cette thèse est à classer parmi cette famille de méthodes, d'où l'intérêt de présenter les techniques existantes en *IBR*.

Après une étude des techniques de *billboarding* en 2.3.1, nous examinerons en 2.3.2 le concept de fonction plénoptique, en 2.3.3 les techniques utilisant l'image et sa carte de profondeur, en 2.3.4 une méthode de Max spécifique aux arbres et nous finirons en 2.3.5 par un bilan des techniques à base d'images.

2.3.1 Billboards et dérivés

Une représentation très utilisée dans les systèmes temps réel comme les simulateurs ou les jeux vidéo est le *billboard*. Un *billboard* est un polygone recouvert par une texture, représentant par exemple un arbre, que le moteur de rendu oriente toujours vers l'observateur (*cf.* figure 2.10 à gauche et à droite).

L'avantage de cette technique est d'offrir un modèle simple à mettre en œuvre et très efficace ; il est alors possible d'avoir de nombreux arbres pour un faible coût. Le manque de réalisme des arbres lorsque l'observateur se déplace librement dans la scène est souvent caché par le fait que la caméra (*e.g.* un avion ou une voiture) passe à grande vitesse à côté des *billboards*. De plus, le coût mémoire augmente vite si l'on veut avoir une diversité d'apparence des arbres. À noter aussi l'impossibilité de changer l'illumination ou l'ombrage d'un tel modèle parce que celle-ci est fixée dans l'image. Une vue de dessus serait également incorrecte puisque les arbres paraîtraient couchés, une variante consiste alors à contraindre l'orientation verticale, puis à effectuer la rotation en gardant autant que posible l'arbre face à la caméra (*cf.* figure 2.10 à droite).

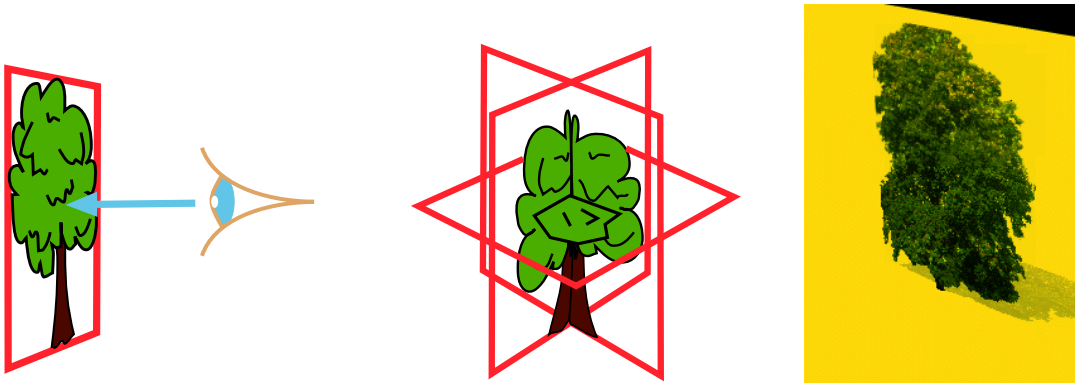

FIG. 2.10 – *À gauche* : *billboard* classique. *Au milieu* : *billboard* croisé. *À droite* : l'aspect plat d'un *billboard*.

Une amélioration possible en terme de réalisme au *billboard* est l'utilisation de trois polygones texturés placés en forme de croix pour représenter un arbre (*cf.* figure 2.10 au milieu), ce qui permet de véritablement tourner autour de l'objet : nous appellerons ceci les *billboards* croisés. Bien qu'améliorant le réalisme des arbres quand on les voit de dessus, cette technique conserve tous les problèmes des *billboards*, comme le peu de réalisme (à cause du manque de relief et de volume), le coût mémoire, avec en plus, le problème technique que pose le mélange des trois images semi-transparente par le moteur de rendu.

Les jeux vidéos utilisent souvent une représentation combinant la géométrie pour le tronc avec des *billboards* pour les branches. Ceci augmente le réalisme des vues proches, mais est plus coûteux en mémoire et en temps de rendu.

Avec le même esprit de mixer géométrie et *billboards* Pulli dans [PCD+97] propose une technique partant d'une série de vues d'un objet dont il dispose d'un maillage grossier. Pour effectuer le rendu depuis un point de vue quelconque, il effectue trois rendus du maillage texturé de l'objet : pour chaque rendu, les textures sont différentes car extraites des séries de vues et seuls les polygones significatifs du maillage sont considérés. Puis, il reconstruit l'image finale en interpolant par pixel les trois images ainsi obtenues.

2.3.2 Fonction plénoptique

En 1995 Bishop [MB95] introduit la fonction *plénoptique* d'Adelson et Bergen dans le but de reconstruire l'image d'un objet depuis n'importe quel point de vue à partir d'une série d'images de cet objet prises lors d'un travelling (*cf.* figure 2.11 à gauche). Cette fonction est à l'origine de deux techniques très connues se basant sur la représentation du champ de lumière, appelé *light field* [LH96] ou *lumigraph* [GGSC96].

Le champ de lumière décrit les caractéristiques complètes de la lumière dans une scène, c'est-à-dire la radiance en un point dans une direction donnée. Ceci est une fonction 5D : position 3D et direction 2D. En obtenant le champ de lumière à partir d'une série d'images, on peut générer de nouvelles images de l'objet sous un nouveau point de vue, simplement en le rééchantillonnant.

À la différence de la plupart des techniques de rendu à base d'images que nous allons voir plus loin, l'image de profondeur n'est pas nécessaire, on peut donc appliquer cette technique à des photographies. Les inconvénients majeurs de ces techniques sont le coût mémoire qu'elles engendrent, la faible possibilité d'accélération du rendu par le matériel graphique standard (à cause de la dimension de la fonction), ainsi que l'impossibilité de changer l'éclairage de la scène puisque celui-ci est stocké implicitement dans les images initiales. En outre, certaines parties cachées de l'objet ne sont pas capturées dans les images, ce qui se traduit lors du rendu par des trous qu'il faut combler en interpolant les valeurs voisines. Il n'est donc pas pensable aujourd'hui d'appliquer une telle technique au rendu de centaines d'objets comme par exemple, une forêt.

Récemment Miller [MRP98] et Wood [WAA+00] ont proposé une technique de compression des *lightfields* couplée à des algorithmes de décompression peu coûteux, limitant ainsi la place mémoire occupée par ces structures de données.

2.3.3 Tranches d'images de profondeur

Shade en 1998 [SGHS98] propose les *Layer Depth Images* (*LDI*), une méthode utilisant plusieurs images et leur tampon de profondeur associé, offrant ainsi une meilleure parallaxe que les techniques vues à la section précédente, et ceci avec moins de données. Cependant cette technique, tout comme les *lumigraphs*, ne permet pas le temps réel sur des centaines d'objets, ni le changement d'éclairage. Les objets semi-opaques ne sont pas représentables, parce que le tampon de profondeur ne stocke qu'une unique valeur de profondeur par pixel. L'année suivante cette technique sera rendue hiérarchique par Bishop dans [CBL99].

Oliveira dans [OBM00] propose une technique dérivée de celle de Shade, mais basée sur la distorsion des textures au moment du rendu pour donner une illusion de perspective.

2.3.4 Un modèle à base d'images de profondeur adapté aux arbres

Max dans [MO95, Max96, MDK99] applique aux arbres les représentations basées sur l'image et son tampon de profondeur. Dans son modèle, un pixel de l'image va contenir la couleur, la profondeur ainsi qu'une unique normale. La représentation est hiérarchique (dans la version la plus récente), c'est-à-dire qu'il part des images d'une feuille (une feuille réelle scannée), avec lesquelles il construit une petite branche dont il tire une image et son tampon de profondeur, lesquels sont utilisés pour construire une branche entière, etc. jusqu'à arriver au niveau de l'arbre. Il pré-calcule une vingtaine de points de vue pour chaque objet, qu'il sélectionne au moment du rendu en fonction de la position de l'œil. Le rendu est effectué avec l'aide du matériel graphique, l'illumination est calculée en post-traitement à l'aide de la normale et de la couleur stockée dans chaque pixel de l'image. L'antialiassage est obtenu par sur-échantillonnage.

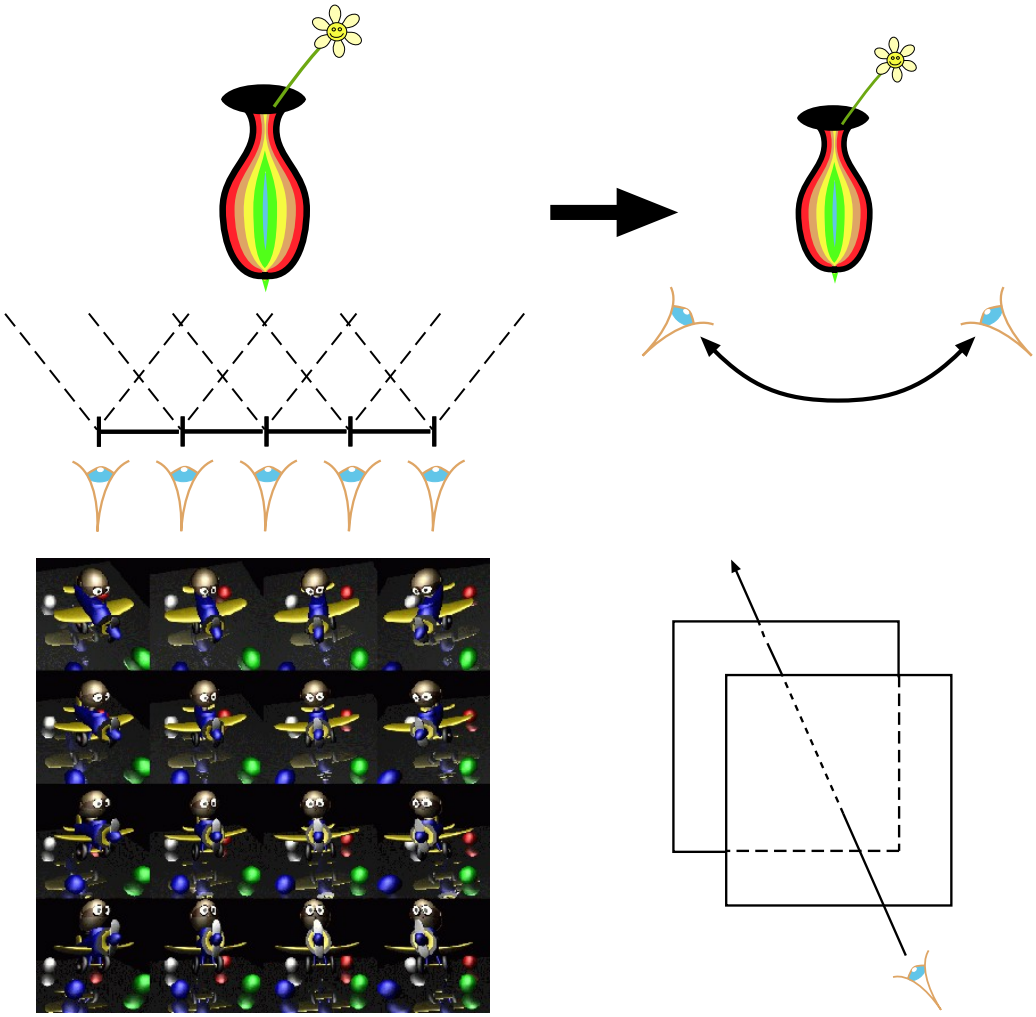

FIG. 2.11 – La technique des *Lightfield* [LH96] ou des *lumigraph* [GGSC96] permet à partir d'une séries d'images (*à gauche*), de reconstruire les points de vue proches d'un objet (*à droite*).

Les inconvénients d'une telle technique sont, comme pour la technique des *LDI* :

- la difficulté d'utiliser pleinement la carte graphique pour accélérer le rendu à cause du traitement final calculant l'illumination,
- l'impossibilité de traiter des objets semi-opaques à cause du tampon de profondeur stockant une unique valeur,
- l'utilisation d'une unique normale par pixel alors que souvent un pixel représente plusieurs feuilles, voir branches.

Max parvient à rendre un très beau verger d'une dizaine d'arbres avec un temps de quelques minutes par images (cependant le matériel actuel devrait permettre d'améliorer nettement ces performances). L'idée de la hiérarchie basée sur les images est très intéressante dans le cas des arbres puisque eux-mêmes sont hiérarchiques. Ce modèle a été une forte source d'inspiration pour celui que nous présenterons dans la partie III de cette thèse.

2.3.5 Bilan des approches à base d'images

Les techniques adaptées au rendu temps réel de grands nombres d'objets, comme les *billboard*, ont le défaut de ne pas offrir un réalisme satisfaisant, de ne pas permettre une grande diversité d'arbres à faible coût mémoire, et de ne pas autoriser la modification dynamique de l'illumination.

La grande qualité des approches *LDI* [SGHS98] ou de celles de Max [MO95, Max96, MDK99] est de pouvoir reconstruire un objet 3D de manière très réaliste à partir de quelques points de vue, par contre l'impossibilité d'utiliser le matériel graphique pour le rendu (ou très peu) est une limitation forte si l'on veut utiliser ces techniques pour rendre une forêt. De plus, il est souvent difficile de modifier l'illumination des objets une fois les images capturées. En revanche, l'idée de Max de calquer une hiérarchie à base d'images sur celle des arbres est très intéressante et doit être retenue.

2.4 Textures volumiques

Le modèle de *textures volumiques* introduit par Kajiya et Kay [KK89] pour le rendu de fourrure puis généralisé par Neyret [Ney95, Ney96, Ney98], est très intéressant sur deux points : tout d'abord parce qu'il constitue une représentation alternative traitant efficacement la complexité de nombreux objets (fil conducteur de ce chapitre), en outre, l'intégrale de l'illumination sur un cylindre effectuée par Kajiya et Kay est le point départ du modèle que présentons au chapitre 4. Nous détaillerons donc le modèle initial en 2.4.1, suivi des améliorations apportées en 2.4.2 et 2.4.3, et nous terminerons par un bilan en 2.4.4.

2.4.1 Le modèle initial [KK89]

Kajiya et Kay développent une approche orientée texture pour représenter des géométries répétitives complexes recouvrant une surface à la manière d'une peau épaisse, comme par exemple la fourrure d'un animal, une forêt sur une colline, etc. Un volume cubique contient un échantillon de référence de cette géométrie encodée sous forme d'un volume de voxels. Les instances de ce volume plaquées sur une surface sont appelées *texels* pour *texture element* (*cf.* figure 2.12). Ce volume de référence est déformé de façon à être plaqué sur la surface, à la manière d'une texture 2D classique.

Le volume cubique de référence est constitué de voxels. Chaque voxel contient formellement trois informations :

- une densité (i.e. une présence) ;
- un ensemble de 3 vecteurs donnant l'orientation locale de la surface (Normale, Tangente, BiNormale) ;
- une fonction indiquant comment la lumière se réfléchit (réflectance).

L'orientation locale de la surface et la réflectance peuvent être regroupées en une unique donnée. Un texel contient donc une information spatiale (densité) et une information sur le comportement local vis à vis de la lumière (réflectance). En réalité, dans leur implémentation, Kajiya et Kay stockent uniquement la densité, car l'orientation et la réflectance sont constantes dans un modèle de fourrure : le volume de référence représente un échantillon constitué de cylindres (les poils), il est ensuite instancié et déformé pour suivre le sens du poil. Ils utilisent comme fonction de réflectance l'intégrale analytique approchée de l'illumination d'un cylindre. L'intégrale du terme spéculaire n'est pas réellement calculée mais est construite de manière *ad hoc*. Plus tard, cette illumination analytique sera améliorée par Goldman dans [Gol97].

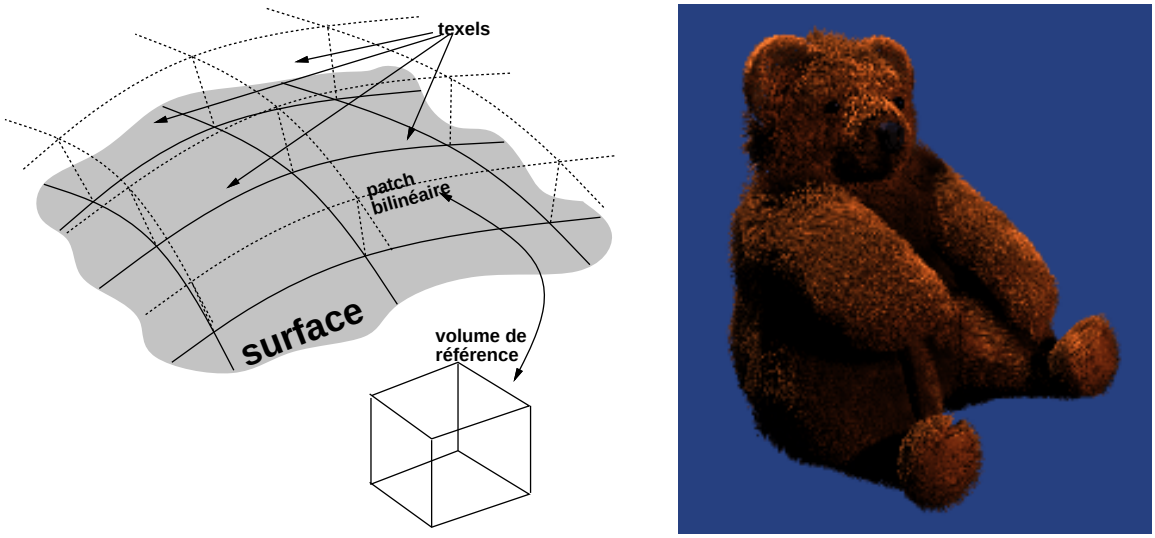

FIG. 2.12 – *À gauche* : le principe des textures volumiques est de plaquer sur la surface un volume de référence pour former une peau épaisse sur l'objet. *À droite* : l'ours de Kajiya et Kay [KK89].

Le volume de référence est destiné à être plaqué de manière répétitive sur toute la surface comme pour une texture surfacique classique. Pour qu'il y ait continuité entre texels voisins il faut que le volume soit déformé : Kajiya et Kay font correspondre les quatre arêtes verticales du volume avec des vecteurs stockés aux nœuds qui sont initialement les normales à la face, mais que l'on peut orienter différemment pour déformer la texture.

Le rendu de Kajiya et Kay utilise un algorithme de lancer de rayons qui devient un peu particulier lorsque celui-ci traverse un texel. Lorsque le rayon traverse un texel on se reporte dans le volume de référence où on applique une technique de rendu volumique : le rayon parcourt les voxels tout en accumulant la transparence et l'illumination locale, qui est évaluée en appliquant la fonction de réflectance, pondérée par l'ombrage obtenu en lançant un rayon entre le voxel et la source de lumière.

Ce modèle a permis à Kajiya et Kay de réaliser de belles images d'ours en peluche, sans aliassage (*cf.* figure 2.12 à droite), mais au prix d'une douzaine d'heures de calcul (à l'époque). Nous allons voir à la section suivante comment cette technique a été étendue et améliorée.

2.4.2 Un modèle plus général

Dans [KK89] les auteurs voulaient représenter une texture bien particulière, la peluche sur un ours. Bien que l'architecture proposée soit générale, le texel et la fonction de réflectance utilisés sont très spécifiques : le texel, qui doit matérialiser un ensemble de poils, contient des cylindres perpendiculaires à la base du volume, et la fonction de réflectance utilisée est cylindrique, n'autorisant que des objets cylindriques dans le volume de référence. Une généralisation de ce modèle a été réalisée par Neyret [Ney95, Ney96, Ney98].

Neyret propose une fonction de réflectance paramétrable (l'ellipsoïde) qui est capable de modéliser de nombreux types de formes. La méthode originale est très lente : malgré l'échantillonnage du rayon, le rendu de Kajiya et Kay considère inutilement beaucoup de voxels dans le cas où

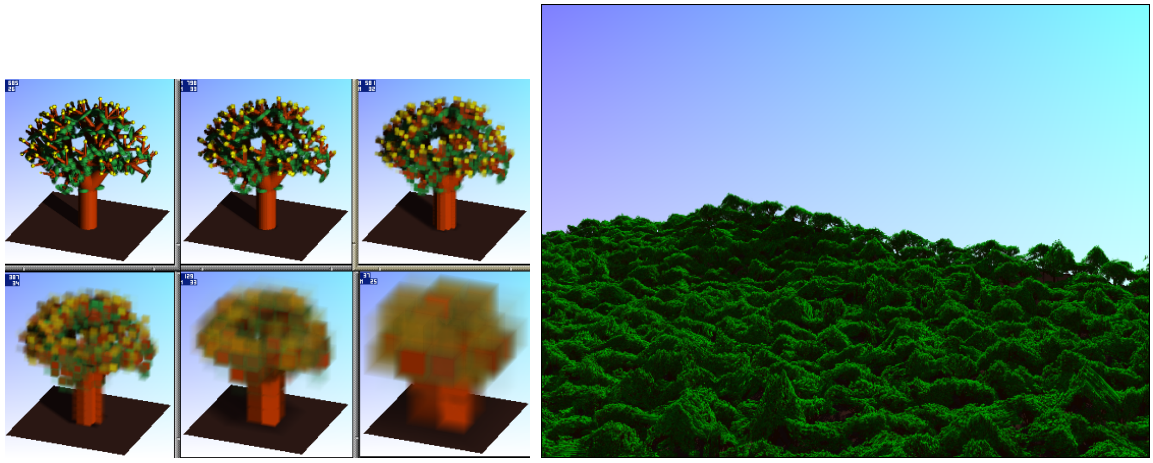

FIG. 2.13 – *À gauche* : un texel représentant un arbre à différents niveaux de détails. *À droite* : une forêt générée par la méthode de Neyret [Ney96], une extension des textures volumiques.

le volume est loin de l'observateur[2]. Neyret introduit donc une approche multi-échelles similaire au *MIP mapping* utilisant les octrees, ce qui n'est possible qu'en introduisant une primitive plus générique ayant une structure de "groupe", *i.e.* qui permette de représenter aussi la combinaison des primitives (*cf.* figure 2.13). Le niveau de détails affiché est ainsi modulé en fonction de la distance qui sépare le texel de l'observateur, ce qui procure un gain de vitesse appréciable, en faisant disparaître les contrastes des détails, mais en gardant la même qualité d'image (i.e. avec très peu d'aliassage). Les octrees apportent aussi un gain en taille non négligeable (le taux de compression est de l'ordre de 95%), en effet un texel non compressé peut vite atteindre une taille importante.

Avec ce modèle enrichi de texture volumique, Neyret arrive à rendre des images de bonne qualité (*cf.* figure 2.13 à droite) dans un temps tout à fait raisonnable, par lancer de rayons (10 à 20 minutes à l'époque) avec un unique rayon par pixel.

2.4.3 Textures volumiques dédiées aux arbres

Noma [Nom95] dérive le principe des textures volumiques de Kajiya pour le rendu spécifique d'arbres. En chaque voxel, il stocke l'opacité de manière discrète en échantillonnant la géométrie depuis une série de directions autour de la sphère, lors du rendu il interpole les valeurs. Pour la fonction d'illumination, il calcule en chaque voxel la moyenne des normales des feuilles et la moyenne des cosinus des angles entre ces normales et les trois axes X,Y,Z. Lors du rendu, il interpole ces valeurs en fonction des angles entre la lumière et les axes X,Y,Z. Il utilise de la géométrie pour les arbres proches qu'il mélange aux texels pour les arbres éloignés.

2.4.4 Bilan des textures volumiques

Les textures volumiques peuvent être vues de deux manières :
- une représentation pour les scènes complexes, facile à contrôler par l'utilisateur ;
- une représentation permettant un rendu très efficace en temps et en qualité, car générique et multi-échelle.

[2] A noter que ce problème existe aussi pour les textures 2D et a été résolu par la technique du *MIP mapping* en pré-calculant la texture à diverses résolutions(*cf.* section 1.1.2).

Cette représentation alternative propose de remplacer une géométrie complexe par un octree où chaque voxel est représenté par une fonction analytique d'illumination. Neyret propose une fonction générique ellipsoïdale. Cette fonction de réflectance générique peut s'avérer inadaptée à certaines géométries : les arêtes vives, ou toutes géométries où la distribution de normales comporte des discontinuités. Pour remédier à ceci l'idée serait de calculer une série de fonctions d'illumination pour des classes d'objets spécifiques : c'est cette idée que nous allons développer durant la partie II.

2.5 Couches d'images

Les textures volumiques donnent des résultats visuels réalistes, et rapidement pour une technique utilisant le lancer de rayons. Cependant, pour des applications exigeant le temps réel, l'obtention d'un gain supplémentaire parait difficilement envisageable en conservant cette technique telle quelle. Des approches adaptées au matériel graphique ont donc été développées. Les moteurs de rendu des cartes graphiques traitant efficacement des polygones, nous présentons ici des techniques basées sur le rendu par couches, où chaque couche est représentée par un polygone texturé. Cette idée était à l'origine destinée au rendu volumique puis elle s'est généralisée à divers domaines du rendu.

Nous verrons en 2.5.1 les techniques de rendu volumique introduisant l'idée de tranches, puis nous détaillerons en 2.5.2 la technique de texture volumique temps réel, ainsi qu'en 2.5.3 les améliorations apportées. Nous finirons en 2.5.4 par un bilan de ces modèles à base de couches.

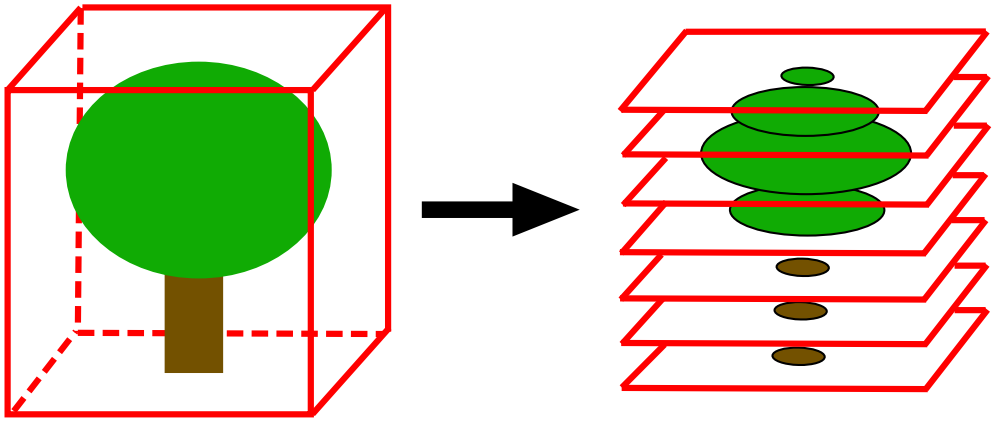

FIG. 2.14 – Le volume de référence des textures volumiques temps réel [MN98] est une superposition de polygones texturés.

2.5.1 Rendu volumique par couche

Le rendu volumique se calculait usuellement en lançant des rayons à travers l'espace voxelisé, ce qui se traduisait par des temps de calcul très longs. Pour l'accélérer Lacroute et Levoy introduisent le rendu par couches d'images [LL94]. Ils interprètent leurs données volumiques comme des couches, avec l'idée de factoriser les voxels en une texture et de traiter en parallèle ces données. Pour le rendu ils proposent de projeter et composer successivement les couches sur le plan image et ainsi obtenir l'image finale. Chaque couche projetée est combinée avec le résultat précédent en tenant compte de la densité, on peut donc interpréter une couche comme une texture transparente. La projection d'une couche est plus rapide que les calculs de projection pour chaque voxel le long d'un rayon : cette factorisation permet donc un gain de temps important. Lacroute et

Levoy ont imaginé une méthode pour projeter les couches orthogonalement quel que soit l'angle sous lequel on regarde le volume, même avec un rendu en perspective.

Westermann et Ertl [WE98] proposent une adaptation de cette technique en profitant des fonctionnalités des cartes graphiques pour effectuer le rendu en temps réel. Leur implémentation permet même de tenir compte de l'illumination mais à cause de contraintes des cartes graphiques seul un rendu monochrome est possible : ils stockent la normale dans les trois composantes *RGB* du tampon. Le calcul de l'illumination se fait en post-traitement sur tout le tampon à l'aide du matériel graphique. Cette technique est distribuée comme extension *OpenGL* sous le nom de *Volumizer* [SGI].

2.5.2 Texture volumique temps réel

Neyret et moi-même [MN98] (correspondant à mon travail de DEA) avons développé une technique temps réel des textures volumiques (*cf.* section 2.4), en nous inspirant du modèle de Lacroute et Levoy décrit à la section précédente. Nous avons profité de la capacité des cartes graphiques à traiter rapidement des polygones texturés. Le principe est de représenter le texel par une superposition de polygones texturés et transparents (*cf.* figure 2.14).

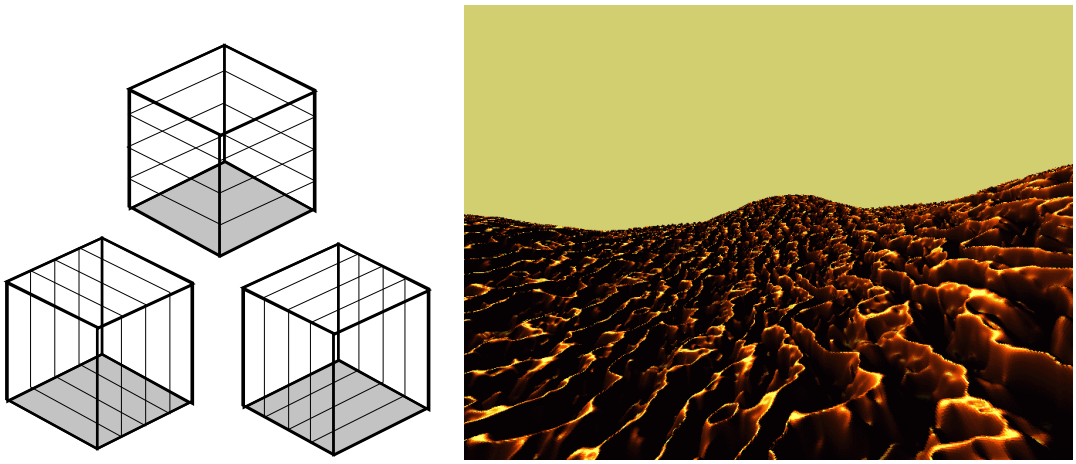

FIG. 2.15 – *À gauche* : trois directions de tranches. *À droite* : exemple de complexité obtenue.

Comme les tranches ne font pas face à l'observateur et n'ont pas d'épaisseur, on risque de voir entre elles. On définie donc trois directions de tranches. Le point de vue de l'observateur détermine laquelle de ces directions va être utilisée (*cf.* figure 2.15). La génération des texels peut se faire de deux manières, soit en convertissant une représentation polygonale, soit à partir d'une texture de hauteur.

Il est à noter que, contrairement à la représentation initiale des textures volumiques, chaque voxel contient directement la couleur de l'objet, c'est-à-dire que l'illumination est inscrite dans le voxel au lieu d'être calculée à chaque rendu. Ceci ne permet donc pas de changements de conditions d'éclairages après la capture des données, mais c'était la concession à faire pour obtenir le temps réel avec le matériel graphique de l'époque.

La même année, la technique de Schaufler [Sch98] utilise aussi des couches d'images pour le rendu accéléré d'objets. Son principe est de regrouper par couches les parties de l'objet ayant la même profondeur, afin de réutiliser cette partie de l'image aux pas de temps suivants (en effet des points se trouvant à la même profondeur se déplacent à la même vitesse quand la caméra se translate).

2.5.3 Autres techniques à base de couches

Modèle dédié aux arbres

Une technique proche de celle des textures volumiques par couches a été proposée dans [Jak00] pour le rendu spécifique d'arbres. Avec leur représentation, seules les feuilles sont représentées dans les tranches de texture, le tronc étant formé par des polygones. L'affichage des trois directions se fait simultanément et non pas en fonction du point de vue comme c'était le cas pour notre modèle. Ils montrent que peu de tranches sont nécessaires au réalisme. Leur modèle peut être rapproché de la technique des *billboards* croisés (*cf.* section 2.3.1).

Modèle dédié à la fourrure

Dans [Len00, LPFH01], Lengyel présente une adaptation intéressante de notre technique pour le rendu temps réel de fourrure. Son volume de référence est constitué de poils. Pour résoudre le problème des vues transversales il n'utilise pas trois directions de tranches comme nous le faisions, mais quatre polygones texturés représentant la silhouette qu'il plaque sur les quatre côtés du texel. Avec cette idée, les comportement des vues aux angles rasants par rapport à la surface deviennent très convaincants pour une approche temps réel (*cf.* figure 2.16 à droite).

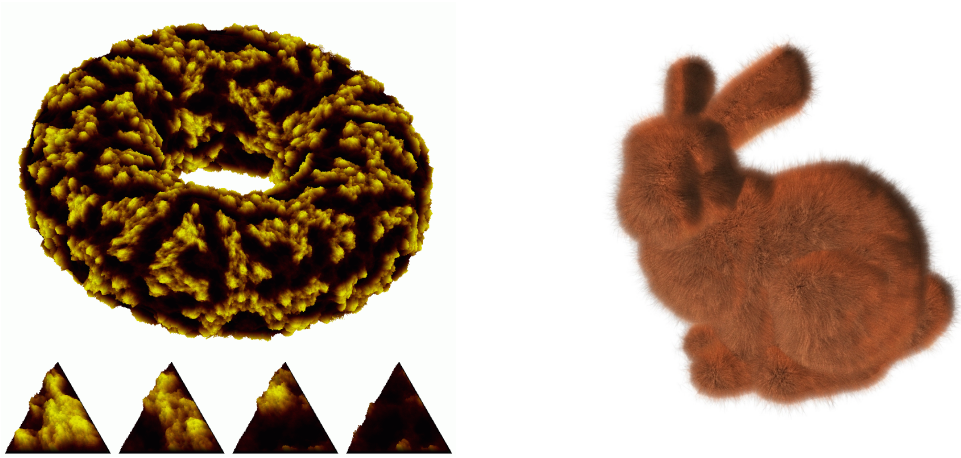

FIG. 2.16 – *À gauche* : un tore recouvert de texels par la méthode de [MN98], mappé sans distorsion apparente ni répétition avec la méthode de [NC99] basée sur des motifs triangulaires complémentaires. *À droite* : un lapin recouvert de texels par la méthode de [LPFH01].

Textures volumiques et illumination

Des travaux très récents de Sénégas [SN01] utilisent les nouvelles fonctionnalités d'illumination par pixel des nouvelles cartes graphiques[3], dans le but d'ajouter un calcul d'illumination aux texels temps réel.

Le matériel graphique des *SGI* que nous avions utilisé en 1998 ne permettaient pas le calcul de l'illumination au niveau des pixels. Celle-ci était évaluée aux sommets des polygones, puis ces

[3]Cartes *GeForce* 2 et 3 de la société *NVidia* et *Radeon* de la société *ATI*

valeurs étaient interpolées lors de la *rasterisation*[4]. Les nouvelles fonctionnalités des cartes graphiques permettent d'évaluer une fonction d'illumination en chaque pixel d'un polygone (*cf.* figure 2.17 à gauche).

Une couche d'un texel est alors composée de deux images : une image de couleur, comme dans notre représentation initiale et une image de normales. Les trois composantes *RGB* de l'image de normales correspondent aux coordonnées de la normale à la surface. Les fonctionnalités dont nous parlions plus haut permettent de micro-programmer une fonction d'illumination en chaque pixel du polygone en tenant compte de la couleur et de la normale. Ceci permet donc de changer interactivement la position de la lumière dans la scène. Il ne manque plus que l'ombrage et l'auto-ombrage pour obtenir un modèle alternatif complet (ce sont d'ailleurs des travaux en cours). Des travaux très récents, proches de ceux-là, ont été développés pour le rendu volumique [EKE01].

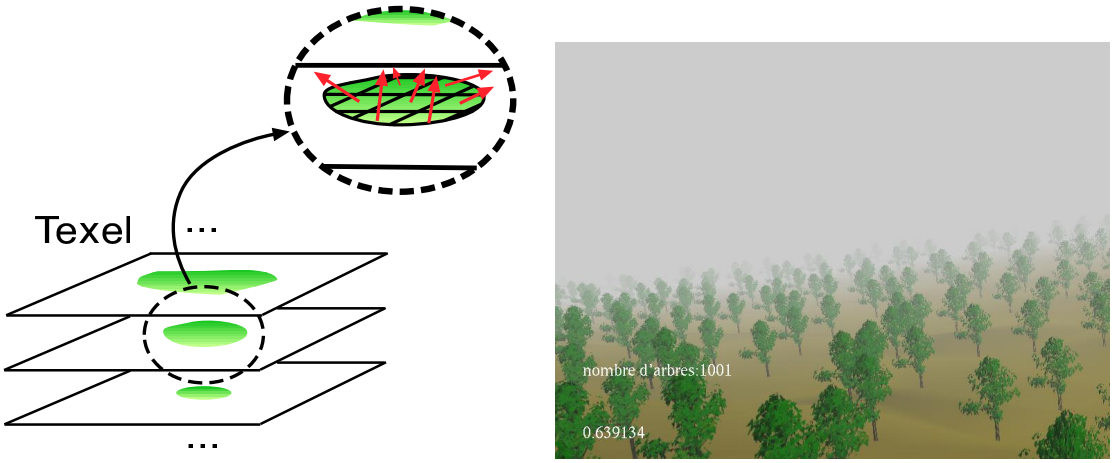

FIG. 2.17 – *À gauche* : avec les nouvelles générations de cartes graphiques, on peut évaluer une fonction d'illumination en chaque pixel du polygone. *À droite* : exemple de rendu temps réel d'une forêt de texel avec l'illumination par pixel [SN01].

2.5.4 Bilan des approches à base de couches d'images

Avec ces approches à base de couches d'images, les coûts de rendu deviennent proportionnels à la complexité de la surface et non à la complexité des objets la recouvrant. Une forêt aura un coût de $n \times m$ polygones où n est le nombre de polygones du terrain et m le nombre de couches pour représenter un arbre (de 64 à 256), ce qui est peu comparativement à sa complexité réelle (représentée visuellement par des milliards de polygones). Pour représenter les micro-géométries, traitées par les cartes de déplacement, il est possible d'obtenir le temps réel avec une technique à base de couches [KS01].

Les fonctionnalités de calcul de l'illumination par pixel des nouvelles cartes graphiques sont en train de lever les limites qu'avaient ces techniques en terme d'illumination. Par contre, la taille mémoire nécessaire à la représentation d'un objet reste toujours une limite à la diversité des objets utilisables simultanément (les performances des cartes graphiques ont explosées, mais pas leur capacité mémoire).

[4]Le processus de *rasterisation* est le fait de remplir le polygone 2D, pixel par pixel, une fois ses sommets projetés sur le plan de l'écran.

2.6 Rendu de macro-géométrie avec une fonction bi-directionnelle de texture

Dischler en 1998 [Dis98] propose une technique de textures de macro-géometrie permettant de traiter simplement des textures en relief comme de l'osier tissé ou des armatures métalliques. Pour cela, il concilie l'idée de lancer de rayons virtuel avec l'utilisation d'une sorte de textures de paramètres de Blinn (*cf.* section 1.1.2).

Le principe de lancer de rayons virtuel est simple : lorsque le rayon arrive sur une surface, on le transpose dans un volume de référence, où le lancer de rayons se poursuit (on retrouve la notion de volume de référence présente aussi dans les textures volumiques).

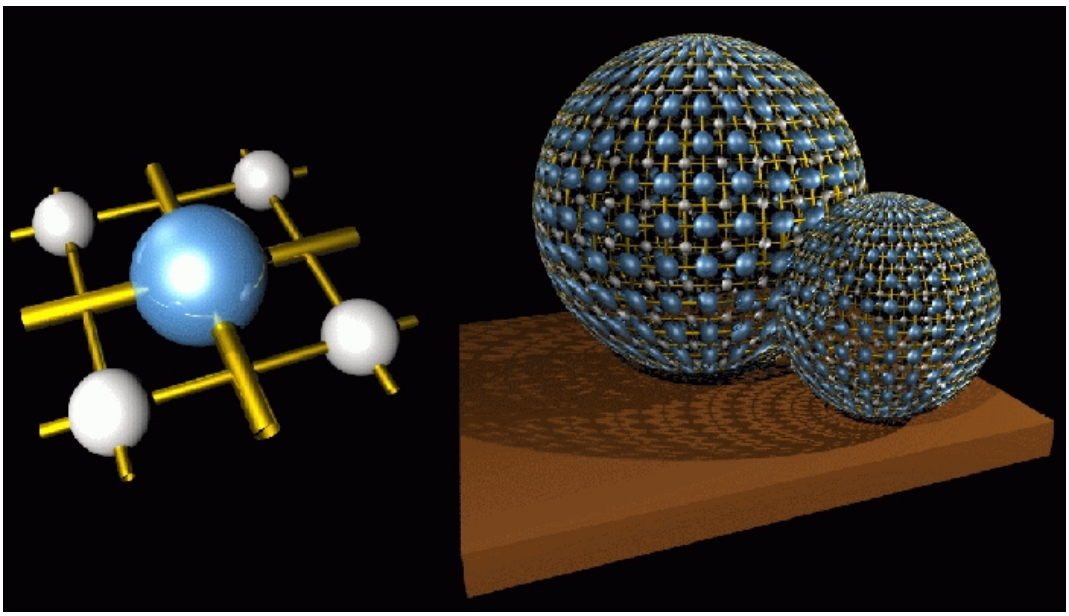

FIG. 2.18 – Les textures de paramètres de Dischler [Dis98]. *À gauche* : le volume de référence. *À droite* : deux sphères formées à partir du volume de référence.

L'idée de Dischler est de stocker dans son volume de référence une fonction de texture bi-directionnelle (*BTF*), qualificatif que Dana avait déjà employé pour ses collections d'images de matériaux prises sous tous les angles (*cf.* section 1.1.7). La texture utilisée est plate, comme une texture ordinaire mais son apparence change avec le point de vue, reproduisant ainsi les effets de parallaxe du relief. Chaque pixel de cette texture stocke des paramètres de normales et de couleurs différents selon la direction d'arrivée du rayon.

Le résultat est très convaincant (*cf.* figure 2.18), d'autant plus que ce type d'objet n'aurait pu être représenté ni par une *bump-map*, ni par une *displacement map*. Il y a cependant des défauts dus à l'absence d'inévitable codage 3D : une surface très distordue présentera des artefacts de parallaxe ; en outre, les rayons ne passent pas d'une case à l'autre, ce qui est pourtant nécessaire dans le cas d'un rayon rasant. Cependant, l'idée de pré-calculer l'information depuis toutes les directions est un principe que nous allons reprendre avec la technique présentée en III.

3 Bilan

À la vue des conclusions partielles que nous avons tirées après la présentation de chaque famille de modèles traitant la complexité géométrique de la nature, il devient clair qu'il n'existe pas

de technique unique et générale résolvant tous les problèmes. L'utilisation de polygones n'étant pas satisfaisante dans de nombreux cas, les chercheurs en synthèse d'images doivent concevoir d'autres représentations plus efficaces, ce qui, comme nous venons de le voir, fonctionne plutôt bien en terme de qualité et de coût. Seulement cette efficacité a un prix : la spécificité. Les modèles alternatifs sont souvent dédiés à une famille d'objets précis, leur domaine d'application est alors plus réduit, ce qui va de pair avec une certaine complexité de mise en œuvre : pour une représentation nouvelle, il faut redéfinir toute la chaîne qui va de la modélisation au rendu en passant par l'animation, et gérer son intégration avec les objets traditionnels[5].

Les paysages, et en particulier les forêts, sont constitués d'objets spécifiques. Cependant l'intérêt en terme d'applications est tel, que le développement d'activités dédiées est justifiable, d'autant que les représentations traditionnelles ne parviennent pas à offrir des solutions acceptables. Ce constat nous a conduit à développer durant cette thèse, des techniques spécifiques aux arbres, s'inspirant de tout ce que nous venons d'exposer ici.

[5] Il existe des projets [Res, Dis, Sof] essayant de regrouper en une application les techniques existantes en matière de synthèse d'images de paysages.