Numéro d'ordre : 203

THÈSE

présentée à

l'Université de Strasbourg – école doctorale MSII
Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection
UMR 7005 CNRS/UDS

par

M. Guillaume GILET

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE STRASBOURG
DISCIPLINE : INFORMATIQUE

AJOUT DE DÉTAILS VISUELS COMPLEXES EN TEMPS RÉEL POUR LA SYNTHÈSE D'IMAGES

---

soutenue publiquement le 24 septembre 2009
devant le jury composé de :

M. Bruno LEVY, Directeur de Recherche INRIA, rapporteur externe
M. Mathias PAULIN, Professeur des Universités, rapporteur externe
M. Jean-Michel DISCHLER, Professeur des Universités, directeur de thèse
Mme. Dominique BECHMANN, Professeure des Universités, examinateur
M. Nicolas HOLZSCHUCH, Directeur de Recherche INRIA, examinateur

Remerciement

En premier lieu, je tiens à remercier les personnes qui, par leur conseils, discussions, ou simplement soutiens ont contribué à la réalisation de ce manuscrit.
Je remercie dŠabord mon directeur, Jean-Michel DISCHLER, pour mŠavoir encadré et accompagné au cours de ces années de thèse. Je remercie également Luc SOLER ainsi que lŠIRCAD pour mŠavoir accueilli et financé lors de la majeure partie de cette thèse. JŠen profite pour remercier lŠéquipe Virtuals pour son enthousiasme et sa convivialité qui ont agrémentés ces premières années.
Merci également aux membres de mon jury dŠavoir accepté de lire et dŠévaluer mon travail de thèse.
Merci à tous mes collègues de lŠéquipe IGG pour leurs conseils et camaraderie, ainsi que pour ces discussions intéressantes et souvent constructives. Merci en particulier à Basile et à Ben pour mŠavoir encouragé à reformuler mes dires et ainsi rendre compréhensible ces travaux.
Finalement, merci à mes amis et à ma famille pour m'avoir soutenu sans faillir au cours de ces longues années. Je remercie en particulier Pierre, Julien et Mathieu pour leurs conseils de relecture et ces instants de détentes et de camaraderie.

Table des matières

Table des figures

Chapitre 1

Introduction

Le domaine de l'informatique graphique est un domaine en plein essor. Les images générées par ordinateur sont devenues indispensables dans de nombreux domaines, comme le cinéma, l'industrie du jeu vidéo, les simulateurs ou encore le domaine médical. Le besoin d'immersion inhérent à ces applications pousse le domaine de la génération d'images de synthèse dans une course vers le réalisme afin de reproduire un monde virtuel aussi complexe que la réalité qui nous entoure. Il est primordial pour atteindre ce réalisme d'enrichir l'aspect visuel des objets synthétiques. La plupart des modèles générés synthétiquement pèchent par leur aspect trop lisse et parfait et nécessitent plus de richesse visuelle afin de s'approcher de ce réalisme.

Pour représenter des objets virtuels, il faut créer un modèle mathématique décrivant les formes de l'objet. Toutefois, la richesse visuelle d'un objet réel ne découle pas tant de sa forme globale que d'une multitude de détails, comme la texture de l'objet, la présence d'une partie abîmée, ou encore un pelage ou une fourrure dans le cas de modèles organiques. L'ensemble de ces petits détails n'est pas nécessaire à la représentation de l'objet dans sa globalité mais reste indispensable pour s'approcher du réalisme. Intégrer chacun de ces détails « explicitement » dans la représentation informatique globale de l'objet est problématique à plus d'un titre : premièrement, le temps nécessaire pour modéliser un objet virtuel risque d'être élevé de plusieurs ordres de grandeur. Deuxièmement, la complexité d'une telle représentation rendra l'objet difficile à manipuler dans le cas d'animations ou de simulations physiques par exemple. De plus, le coût de stockage engendré par la modélisation de l'ensemble des détails pourra rapidement devenir prohibitif, voire dépasser totalement la capacité mémoire disponible. Dernièrement, il deviendra difficile de faire évoluer l'apparence visuelle d'un objet modélisé de cette manière, ou d'attribuer une apparence arbitraire à un objet, imposant une contrainte non négligeable sur la liberté de création et la diversité des images virtuelles.

La distinction entre la forme d'une surface et les petits détails visuels permet d'introduire un niveau de représentation intermédiaire indépendant de la forme de l'objet : la « mésostructure ». Ce niveau concerne les petits détails visuels qui ne sont pas nécessaires à la compréhension de la forme globale de l'objet mais qui contribuent à son enrichissement visuel. Ce travail de thèse se focalise sur les techniques permettant la représentation et l'affichage de ce niveau de détail. De nombreuses applications peuvent bénéficier de l'utilisation du niveau de mésostructure. Par exemple, une mésostructure représentant un réseau veineux peut venir accroître le réalisme d'un modèle médical ou la représentation d'une fourrure augmenter l'apparence d'un modèle d'animal dans le cadre d'une application cinématographique ou d'un jeu vidéo.

Nous proposons dans cette thèse une méthode permettant l'affichage interactif d'une mésostructure arbitraire sur un objet. Ceci permet d'enrichir la qualité visuelle d'un modèle sans sacrifier totalement les performances d'une application interactive. Nous proposons également quelques outils permettant la modélisation automatique et manuelle des informations de mésostructure.

1.1 Problématique

Afin de différencier les niveaux de représentation d'un objet, nous reprenons la classification introduite par Westin [162] en la complétant avec un niveau intermédiaire entre la macrostructure et la millistructure : la mésostructure, comme illustré par la figure 1.1. La macrostructure d'un objet est la représentation ayant l'échelle la plus grande et représente la forme globale d'un objet. Nous complétons cette classification avec la mésostructure, qui n'est pas en elle-même nécessaire à l'identification de l'objet (un vase reste un vase, indépendamment de l'état ou des motifs de sa surface) mais permet d'enrichir visuellement le modèle. Westin définit la millistructure pour décrire la structure interne d'un matériau (par exemple les fibres d'un vêtement). Il est à noter que ce niveau structurel peut être inclus dans la mésostructure. La microstructure modélise la structure à l'échelle microscopique de l'objet, et est généralement non visible à l'œil nu. La microstructure est toutefois capitale pour le réalisme car elle détermine le comportement optique des matériaux de l'objet, c'est-à-dire la manière dont la lumière interagit avec l'objet. C'est ce niveau structurel qui définit l'apparence finale des éléments de l'objet, qui peuvent par exemple être mats, brillants ou même translucides.

Il est à noter que cette classification n'est pas dépendante d'une métrique à proprement parler mais plutôt de l'échelle des phénomènes observés. Par

exemple, pour une représentation d'un paysage par une vue aérienne, une entière forêt peut être considérée comme un détail de mésostructure tandis qu'un arbre en lui-même relèvera de la microstructure. De manière similaire, lors de l'observation du système solaire, ce sont les continents entiers qui appartiennent à la mésostructure de la représentation du globe terrestre.

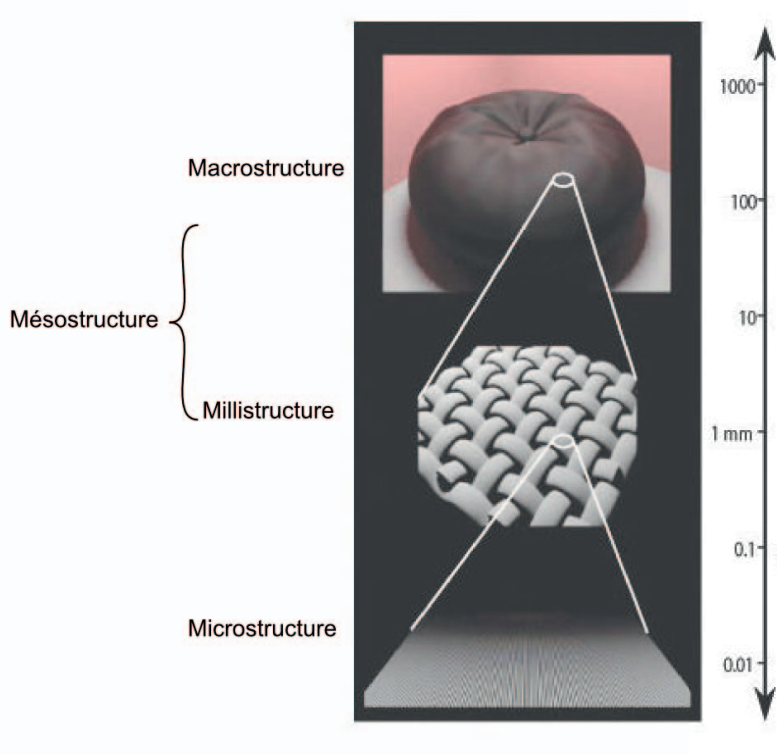


FIGURE 1.1 – Classification par Westin [162] des différentes échelles de représentation d'un objet. Nous y ajoutons le niveau de mésostructure, entre macro et millistructure.

Nous nous focalisons dans cette thèse sur la mésostructure uniquement, négligeant les niveaux macro- et microstructurels d'un objet. La problématique principale de ces travaux est d'enrichir la complexité visuelle des modèles synthétiques par le biais de mésostructure dans le cadre d'applications diverses. Cela pose plusieurs contraintes :

Interactivité Les besoins des acteurs du domaine graphique sont en constante évolution. Afin d'utiliser ces modèles dans le plus large contexte possible, notamment dans le cadre d'applications interactives, comme les jeux vidéos ou les simulateurs, les performances (en terme de vitesse d'affichage) d'une technique de rendu (d'affichage) de mésostructure doivent

être les plus élevées possible. Cette interactivité est de plus particulièrement importante du point de vue de la création de contenu et de la conception de prototypes. Elle permet un gain de temps considérable pour les artistes et réalisateurs (dans le cadre de l'industrie du cinéma) qui peuvent tester et ajuster immédiatement les effets visuels introduits par la mésostructure.

Facilité de modélisation Cette contrainte concerne les efforts nécessaires pour obtenir une représentation des effets visuels de mésostructure. La mésostructure doit pouvoir être modélisée de manière rapide par les acteurs du domaine de l'industrie graphique. Il est également nécessaire de fournir au modeleur de mésostructure des outils permettant un contrôle précis et efficace sur les résultats visuels créés.

Coût mémoire La multitude de détails responsable de la richesse visuelle d'un modèle pose également la contrainte du coût mémoire de la représentation. En effet, une représentation informatique de mésostructures doit être assez compacte pour pouvoir être utilisée efficacement.

Indépendance de la mésostructure Une contrainte supplémentaire est de respecter l'indépendance de la mésostructure par rapport aux autres niveaux structurels. Une méthode ajoutant des détails de mésostructure doit être capable de s'abstraire des autres niveaux structurels et permettre l'ajout de détails de mésostructure indépendamment de la description mathématique d'un objet. Cette indépendance permet également l'application de détails arbitraires de mésostructure sur un modèle. Une application utilisant le rendu de mésostructure doit donc pouvoir s'intégrer facilement dans les moteurs de rendu actuels.

Généricité de la méthode Alors qu'il existe plusieurs méthodes ad hoc pour différentes représentations de mésostructure, l'un des enjeux d'une technique est de pouvoir s'appliquer à une gamme de mésostructure la plus large possible.

Notre objectif est de proposer une méthode de rendu de mésostructure permettant d'enrichir l'apparence visuelle d'un modèle synthétique. Cette méthode doit permettre un affichage rapide et être compatible avec les moteurs de rendu actuels. Nous souhaitons également que la méthode proposée puisse s'adresser à une large gamme de mésostructures et s'appliquer à des modèles arbitraires. Finalement, nous souhaitons proposer des outils permettant une modélisation de mésostructure rapide laissant un contrôle efficace à l'utilisateur final. Nous nous attachons également à concevoir notre méthode afin qu'elle puisse tirer parti des dernières avancées de matériel graphique, condition *sine qua none* pour accéder à l'interactivité.

1.2 Contributions

Nos contributions consistent principalement en une nouvelle approche hybride visant à ajouter des détails visuels de mésostructure à une scène synthétique afin de permettre d'accroître la richesse visuelle des modèles au cours d'une visualisation interactive. Nous proposons dans ce document plusieurs contributions :

Une méthode hybride de rendu de mésostructure Nous proposons une méthode hybride générale autorisant l'ajout de détails de mésostructure à un objet. Cette méthode repose sur l'utilisation conjointe du principe de rasterisation et de lancer de rayons. Elle est également générique et indépendante de la modélisation des détails visuels de mésostructure. Son principal atout est de procéder à l'enrichissement visuel uniquement pour les parties visibles d'un objet. Cette méthode est au cœur de nos travaux et nous montrons son intérêt au travers de plusieurs applications.

Le traitement d'objets volumiques et surfaciques Une des contributions importantes de cette thèse est de permettre indifféremment l'ajout de détails visuels sur des objets définis de manière volumique ou surfacique.

Une gamme de mésostructure étendue Beaucoup de méthodes d'ajout de détails reposent sur un ensemble d'hypothèses simplificatrices réduisant la gamme de détails visuels pouvant être représentés. Notre méthode permet un rendu d'une grande gamme de détails visuels très différents et reste indépendante de la représentation de la mésostructure.

Nous choisissons de montrer l'utilisation de cette méthode au travers de plusieurs applications apportant des contributions dans divers domaines de l'informatique graphique :

Un outil de visualisation volumique Nous montrons une application de notre méthode de rendu de mésostructure dans le cadre du rendu volumique. Le but de cette application est d'accroître les performances des méthodes de rendu de modèles volumiques non structurés. En effet, les données volumiques résultent de mesures et ne font pas de distinctions entre macro et mésostructure. Le principe de notre application est d'introduire ce niveau de détails en découplant le modèle en deux parties : la macrostructure et la mésostructure. Pour parvenir à un découpage efficace,

nous proposons un critère de définition de mésostructure pour un modèle volumique. Ce découpage du modèle permet de bénéficier des avantages d'une représentation à plusieurs niveaux, comme par exemple une complexité réduite. L'utilisation de notre méthode de rendu résulte en des performances accrues par rapport aux méthodes existantes.

Un outil de modélisation et de visualisation de détails procéduraux L'outil interactif que nous présentons permet la modélisation intuitive de mésostructures volumiques par perturbation de l'espace. Cette méthode est basée sur le principe des hypertextures [123], qui est une technique de déformation spatiale à base de turbulences. L'idée est d'exprimer une mésostructure par un jeu de paramètres, qui seront utilisés lors du rendu pour recalculer une mésostructure « similaire » à celle souhaitée. Il est toutefois souvent difficile de contrôler de manière précise le résultat final, surtout pour des artistes sans expérience de programmation. Nous proposons dans cette thèse une reformulation de la méthode des hypertextures se basant sur une fonction de transfert de forme. Cette fonction, créée à l'aide d'un simple outil de dessin, accroît le contrôle sur l'apparence finale de la mésostructure. Notre méthode de rendu de mésostructure offre de plus un retour interactif à l'utilisateur, améliorant ainsi la facilité de modélisation.

Une génération procédurale de détails colorés surfaciques Finalement, nous proposons une méthode automatique permettant la génération procédurale d'informations de couleur à la surface d'un objet en se basant sur un exemple. Cette technique procède à l'analyse d'une image représentant une texture du monde réel, et procède à la génération automatique d'un jeu de paramètres et d'échantillons de l'image. Lors du rendu d'un objet, ces paramètres sont traités pour recréer pour l'objet une apparence globale « similaire » à l'image d'entrée. Une analyse statistique des caractéristiques visuelles saillantes de l'image d'entrée permet de plus d'augmenter cette apparence en reproduisant (de manière statistique) ces détails visuels particuliers sur la surface de l'objet lors du rendu. Cette méthode permet la génération interactive d'informations de couleur sur des surfaces de taille arbitraire (même infinie) pour un coût mémoire quasi-inexistant.

1.3 Contexte du travail de thèse

Une grande partie de ces travaux de thèse ont été réalisés au sein de l'équipe Virtuals, dirigée par Luc Soler, de L'Institut de Recherche contre le Cancer de l'Appareil Digestif (IRCAD). L'IRCAD, fondée en 1993 par le Professeur Jacques Marescaux, dédiée à la valorisation de la recherche fondamentale contre le cancer et a depuis atteint une réputation d'excellence dans la recherche fondamentale et appliquée, ainsi que dans l'enseignement des nouvelles technologies chirurgicales. De manière plus spécifique, l'équipe Virtuals s'attache à faciliter les procédures chirurgicales guidées par ordinateur en développant l'automatisation chirurgicale, au travers d'applications telles l'analyse automatisée d'images médicales, la modélisation géométrique, topologique et physique des patients en 3D, ainsi que le développement d'outils de planification chirurgicale et de simulation préopératoire et d'aide intra-opératoire grâce à la réalité augmentée. Les travaux présentés au cours du chapitre 6 s'inscrivent dans cette problématique en facilitant l'affichage de grand volume de données volumiques inhérents au domaine de la visualisation médicale. Les deux chapitres suivants abordent de manière plus pronconcée le problème du réalisme visuel, afin de fournir des méthodes capables d'accroître le réalisme lors du rendu d'organes au sein de simulateurs interactifs chirurgicaux. Toutefois, ces méthodes restent très génériques et s'adressent à toutes sortes d'applications de l'informatique graphique. Afin de mettre en évidence cette généricité, nous avons de ce fait pris le parti d'illustrer les méthodes proposées dans cette thèse avec un ensemble de textures naturelles complexes.

1.4 Organisation du document

Nous commençons en première partie du document par présenter quelques éléments nécessaires à la représentation et à la création d'images de synthèse. Cette partie s'organise en trois chapitres. Le premier propose un descriptif de quelques éléments de base utilisés dans le domaine graphique permettant de représenter des modèles de manière informatique (chapitre 2). Le deuxième chapitre passe en revue les principales techniques utilisées pour l'affichage interactif de ces modèles (chapitre 3). Le troisième chapitre est plus spécifique et concerne les différentes techniques présentes dans la littérature traitant du rendu de mésostructure (chapitre 4).

Les parties suivantes du document concernent nos contributions à proprement parler. Nous commençons par présenter notre propre méthode hybride générique (chapitre 5). Les trois chapitres suivants concernent l'application de

cette technique de rendu à diverses problématiques. Nous commençons par préciser l'apport possible de notre méthode sur une application de rendu volumique en procédant à un découplage des données volumiques (chapitre 6). L'avant-dernier chapitre se concentre sur la présentation de notre technique de modélisation et de rendu de mésostructures définies par des hypertextures (chapitre 7). Finalement, le dernier chapitre présente un modèle spécifiquement dédié à la génération procédurale interactive de détails de mésostructure en se basant sur un exemple (chapitre 8). Cette approche permet de générer une information de couleur à la surface d'un objet arbitraire, créant ainsi pour un modèle une apparence voisine d'un exemple donné en entrée par l'utilisateur.

Chapitre 2

Différentes représentations des informations

Pour pouvoir définir une scène synthétique, il est nécessaire de pouvoir la représenter, c'est-à-dire de disposer d'une représentation informatique permettant la description complète des objets la composant. Cette représentation doit, idéalement, permettre la description exhaustive des propriétés des objets, être la plus compacte possible et surtout être suffisamment simple pour permettre un affichage rapide. En pratique, un objet est représenté par un ensemble de primitives, permettant généralement la description de la géométrie, et par un jeu de propriétés, usuellement utilisé pour reproduire les phénomènes et interactions lumineuses complexes prenant place à une échelle plus petite.

Dans cette section, nous présentons les modèles d'objets 3D les plus utilisés en informatique graphique. Nous classifions les différentes représentations par leur type (volumique ou surfacique) et par la définition mathématique de leurs primitives. Il existe une grande variété de modèles permettant de représenter diverses informations, telle la topologie par exemple.
Nous présentons dans ce chapitre un sous-ensemble non exhaustif des représentations les plus utiles à nos travaux (des primitives paramétriques, implicites ou explicites). Nous renvoyons le lecteur vers [14] pour plus de détails sur la modélisation géométrique.
Dans le monde réel, les objets ont un volume et remplissent une partie finie de l'espace. De ce point de vue, une représentation volumique est naturelle. Toutefois, le système perceptif humain perçoit majoritairement des surfaces. Il est de plus suffisant de considérer le bord d'un objet volumique pour obtenir une description de sa forme. Il est donc naturel de s'intéresser également à la représentation de surfaces. Nous commençons ce chapitre par la présentation de quelques-unes des différentes représentations surfaciques (section 2.1) avant de nous intéresser aux objets volumiques (section 2.2). Nous présentons briè-

vement par la suite les modèles d'interaction d'un objet avec la lumière (section 2.3) avant d'aborder le principe de fonction d'attribut permettant de définir divers paramètres pour chaque point d'un objet (section 2.4).

2.1 Représentation surfacique

Une représentation par surface se concentre sur la représentation de l'interface (le bord) entre un volume (un objet) et l'environnement extérieur. Une telle surface est une variété de dimension 2 de l'espace à 3 dimensions. Il est souvent nécessaire pour représenter des surfaces complexes de les décomposer en un ensemble de primitives plus simples. Les primitives peuvent être classées par leur définition mathématique. Nous allons considérer dans un premier temps les primitives paramétriques, avant de continuer sur les primitives implicites, puis explicites.

2.1.1 Primitives paramétriques

Les primitives paramétriques définissent une relation entre les espaces de dimensions 2 et 3. Un point 3D situé sur une surface paramétrique peut être exprimé par des coordonnées (u, v) :

$$\begin{array}{rcl} f : \mathbb{R}^2 & \rightarrow & \mathbb{R}^3 \\ (u,v) & \mapsto & \big(x(u,v), y(u,v), z(u,v)\big) \end{array} \tag{2.1}$$

avec x, y et z des fonctions de $\mathbb{R}^2$ dans $\mathbb{R}^3$.
En règle générale, ces fonctions sont des polynômes de degré trois ou quatre. Les fonctions d'ordre *cubique* sont utilisées majoritairement car elles offrent à la fois une certaine flexibilité et une relative simplicité de calcul.

2.1.2 Primitives implicites

Une primitive implicite effectue la description indirecte d'un volume. Il est toutefois courant de l'utiliser lors du rendu de surface en considérant la surface infinitésimale entre l'intérieur et l'extérieur du volume. Une fonction implicite est une fonction de $\mathbb{R}^3$ dans $\mathbb{R}$, c'est-à-dire qui associe à chaque point de l'espace 3D une valeur scalaire. En règle générale, une valeur négative (resp. positive) signifie que le point de l'espace appartient (resp. n'appartient pas) au volume défini par la fonction implicite. La surface peut alors être décrite par l'équation $f(,x,y,z) = 0$.
En règle générale, les surfaces implicites sont définies sous forme polynomiale.

Un exemple courant de fonction implicite largement utilisée est la famille des quadriques. Elles définissent des formes courantes comme les sphères, les cylindres, les ellipsoïdes et les paraboloïdes hyperboliques et sont représentées par dix coefficients :

$$f(x,y,z) = Ax^2+2Bxy+2Cxz+2Dx+Ey^2+2Fyz+2Gy+Hz^2+2Iz+J = 0 \quad (2.2)$$

Il existe d'autres familles d'ordre supérieur, telles les cubiques ou les quartiques (voir figure 2.1). Ces primitives permettent une représentation extrêmement compacte et précise des surfaces.

FIGURE 2.1 – Des exemples de surfaces implicites d'ordre supérieur. De gauche à droite : une cubique (image de [35]), une quartique, et une sextique (image de [70]).

2.1.3 Primitives explicites

Le type de primitive le plus utilisé pour le rendu de surface est la primitive explicite. Une primitive explicite est définie directement par ses coordonnées dans l'espace 3D. L'exemple le plus courant est d'utiliser comme primitive un polygone planaire. Un polygone est défini par un ensemble de sommets et d'arêtes et permet une représentation polygonale d'une surface. Un maillage polygonal est constitué d'un ensemble connecté de polygones et permet de représenter la plupart des surfaces sous une forme discrète. Pour plus de simplicité, nous nous intéressons uniquement dans cette thèse aux maillages polygonaux représentant une variété de dimension 2. Cela implique quelques restrictions sur les propriétés du maillage. Le type de maillage le plus utilisé est le maillage triangulaire, c'est-à-dire un maillage composé uniquement de triangles. Cela présente plusieurs avantages, tels la convexité des primitives ou l'interpolation barycentrique à l'intérieur du triangle des propriétés définies

aux sommets. De plus, il est aisé de convertir un maillage polygonal en maillage triangulaire (tout polygone peut être triangulé [11]).

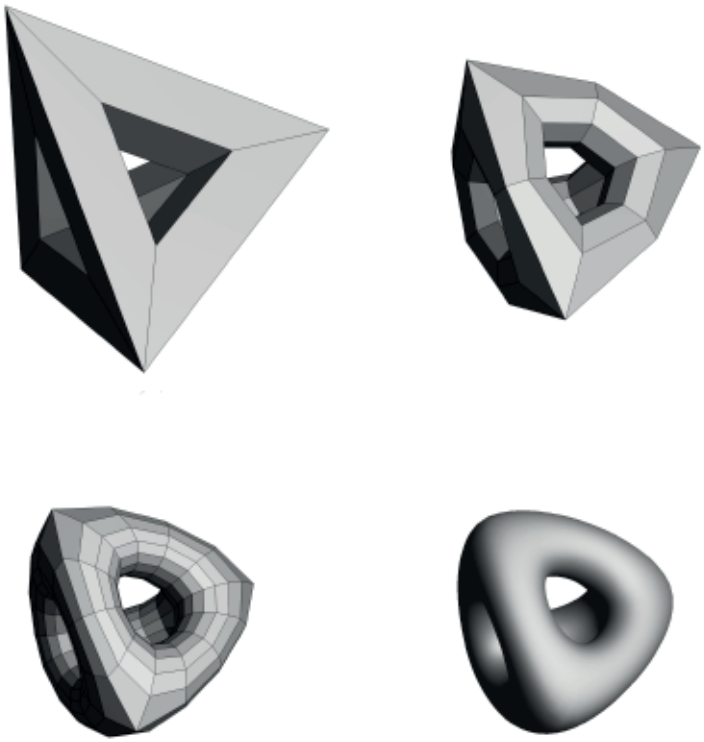


FIGURE 2.2 – Les surfaces de subdivisions sont créées à partir d'un maillage de départ et d'un schéma de subdivision. La surface lisse finale est considérée comme la limite du procédé itératif de subdivision.

Cette représentation par facettes peut être considérée comme une approximation linéaire par morceau d'une surface (du 1^{er} degré). Elle nécessite un grand nombre de polygones pour représenter fidèlement une surface lisse. Une approche permettant la représentation de surfaces sans augmenter le nombre de primitives consiste à utiliser un maillage polygonal en conjonction avec une fonction d'interpolation ou d'approximation d'ordre supérieur. Les primitives du maillage polygonal deviennent dans ces cas les *points de contrôle* d'une surface paramétrique d'ordre supérieur (*surface spline*) (un *patch*) et permettent la

représentation de surfaces lisses sans augmentation du nombre de primitives du maillage.

Cependant, chaque patch de surface est défini de manière locale. Cela implique pour des surfaces à la topologie complexe l'utilisation de plusieurs patchs disjoints. De plus, il est difficile de garder le côté lisse d'une surface tout en raccordant ces patchs disjoints. Ces problèmes sont résolus par une approche de subdivision, permettant en outre la représentation de surfaces lisses par un maillage polygonal. Ces surfaces de subdivision permettent la représentation de surfaces lisses en partant d'un maillage grossier initial [23][42]. Le principe est de modéliser une surface lisse par un maillage polygonal et une règle de subdivision. La surface lisse peut être obtenue en subdivisant le maillage suivant le schéma associé (voir figure 2.2). Le nouveau maillage obtenu est plus proche de la surface et peut à nouveau être subdivisé. La surface lisse finale peut dans ce cas être considérée comme la limite du procédé itératif de subdivision de chaque face polygonale en un sous-ensemble de faces approchant mieux la surface lisse finale. Les surfaces de subdivision sont très utilisées dans l'industrie pour leur simplicité et leur facilité d'édition (il suffit de modifier le maillage grossier d'origine pour déformer la surface).

Les représentations à base de triangles atteignent un certain équilibre entre coût de calcul et capacité à représenter la réalité [145]. Toutefois, pour représenter des surfaces très complexes (comme par exemple un modèle organique), le nombre de triangles nécessaires peut être extrêmement élevé (plusieurs dizaines de millions de polygones). Des travaux [57] [90] ont été menés dans le but de fournir une représentation alternative pour la description de surfaces complexes : la représentation à base de points. Cette représentation s'appuie sur le concept d'utilisation d'un point comme primitive explicite. Ces représentations à base de points (ou de particules) ont été initialement utilisées pour décrire des objets sans géométrie apparente, comme des nuages ou du feu [130].

Zwicker et al. [176] introduisent le *surfel* (de l'anglais *surface element*) pour désigner un élément de surface utilisé dans leur modèle. Ces éléments sont parfois appelés *splats* dans le domaine, en relation avec l'algorithme de rendu associé : le *splatting* (voir section 3.3). Cette représentation repose sur un ensemble d'échantillons situés sur la surface [1] et sans information de connexité entre les éléments (contrairement aux triangles). Chaque échantillon est associé à une fonction de reconstruction qui permet une approximation locale des

1. Ces points peuvent résulter d'un échantillonnage d'une surface quelconque [2] ou d'une acquisition par un scanner 3D.

propriétés de la surface en ce point [124]. Les propriétés de chaque point de la surface peuvent être reconstruites en composant ces fonctions de reconstruction, comme illustré par la figure 2.3. Dans le voisinage d'un point Q de la surface, une paramétrisation locale de la surface est construite en pondérant la projection u_k des échantillons P_k du voisinage sur le plan tangent à Q, en utilisant des gaussiennes radialement symétriques tronquées comme fonctions de reconstruction r_k. La matrice de variance de chaque gaussienne est choisie de manière à refléter la surface autour du point Q. Ces représentations à base de

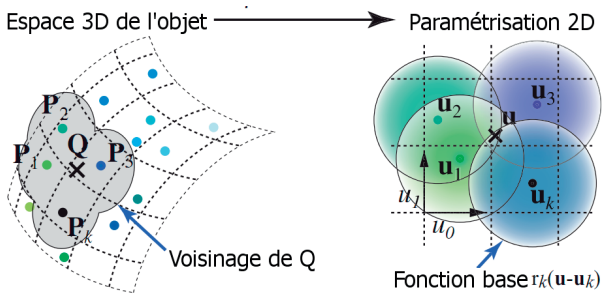


FIGURE 2.3 – A partir d'une représentation à base de surfels, il est possible de reconstruire les propriétés d'un point Q de la surface par une somme pondérée des fonctions r_k de reconstruction des échantillons P_k voisins. *Image de [131].*

points donnent une reconstruction fidèle et efficace de surfaces lisses et permettent souvent un rendu de haute qualité. De plus, beaucoup de systèmes d'acquisition 3D génèrent des nuages de points. De ce point de vue, le point reste une primitive explicite pertinente pour la représentation de surface et représente une alternative viable aux modèles polygonaux.

Une autre représentation explicite souvent utilisée est la structure de *carte de hauteurs* (appelée aussi souvent *champ de hauteurs*). Cette représentation est principalement utilisée pour la représentation de terrain mais peut être également appliquée à la représentation de mésostructures comme nous le verrons au chapitre 4. Il s'agit d'une version discrète d'une fonction qui associe à chaque couple (x, y) une hauteur z. Cette représentation, qui peut être vue comme une carte d'élévation (voir figure 2.4), est compacte car stockée dans une grille 2D avec un scalaire (l'information de hauteur) stocké à chaque échantillon. L'intérêt principal des cartes de hauteurs provient principalement de cette définition compacte, malgré ses restrictions inhérentes. En effet, avec une telle structure de données, il n'est pas possible d'avoir plus d'une hauteur par échantillon, ce qui ne permet pas de représenter des caves ou des trous dans le modèle (cette représentation est appelée *2D et demie*).

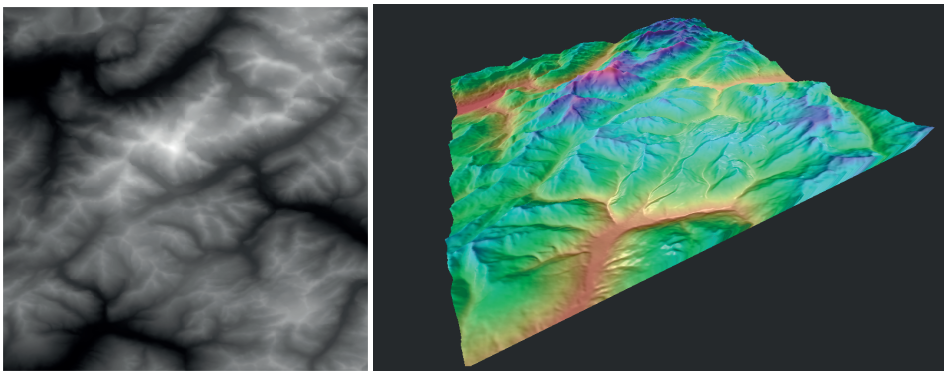


FIGURE 2.4 – Une carte de hauteur (à gauche) associe à chaque couple (x, y) une élévation selon une surface de référence permettant de représenter une surface (à droite).

2.2 Représentation volumique

Une représentation volumique s'attache à définir une présence ou densité de matière en chaque point de $\mathbb{R}^3$. Toute fonction de $\mathbb{R}^3$ dans $\mathbb{R}$ associant une densité à tout point de l'espace peut être considérée comme une représentation volumique. En pratique, une représentation volumique se base sur un ensemble de primitives polyédriques simples (comme le tétraèdre, le cube, l'hexaèdre...) possédant un attribut de densité en chaque sommet et connectés implicitement ou explicitement entre eux. Afin de permettre la reconstruction d'un champ de densités continu sur $\mathbb{R}^3$, il est nécessaire d'associer une (ou plusieurs) fonction d'interpolation ou de reconstruction. La densité de chaque point de l'espace peut donc être exprimée par une combinaison des fonctions de reconstruction de l'ensemble des échantillons pondérés par la densité de l'échantillon associé [2]. La figure 2.5 propose un ensemble des fonctions de reconstructions les plus courantes. La nature de ces fonctions est cruciale pour la qualité de reconstruction des données. Le coût nécessaire à l'évaluation de ces fonctions est également un paramètre à prendre en compte. De nombreux travaux [103] s'intéressent à l'évaluation de la qualité de reconstruction de ces filtres mais ceci dépasse le cadre de cette thèse.

2. En pratique, seul un voisinage des échantillons est considéré plutôt que la totalité des échantillons

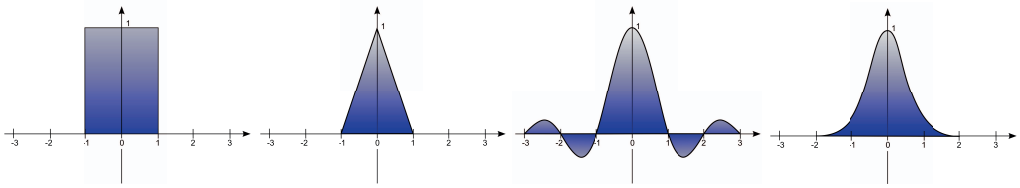


FIGURE 2.5 – Fonctions de reconstruction. De gauche à droite : les filtres boîte, tente, sinc et gaussien.

Il a été montré [8] que le noyau de reconstruction idéal est basé sur le filtre sinc (voir figure 2.5). Toutefois, ce filtre oscille autour de 0 sur l'étendue du domaine de définition. En pratique, des filtres plus simples comme les filtres « boîte » ou « tente » sont utilisés. Il est à noter qu'une représentation à base de *voxels* (de l'anglais *volume element*) peut être considérée comme des échantillons (disposés de manière régulière dans l'espace) associés à une fonction de type « boîte » (à gauche de la figure). L'interpolation linéaire, basée sur le filtre « tente » est souvent utilisée de par son implémentation matérielle dans les cartes graphiques actuelles. Finalement, le filtre gaussien (à droite) offre une alternative intéressante au filtre sinc. Il permet une reconstruction plus lisse qu'un filtre « tente », est moins sujet aux artefacts de discrétisation et reste souvent utilisé dans les représentations volumiques explicites, malgré son coût d'évaluation non négligeable.

Une des primitives couramment utilisée est le voxel décrivant un élément de volume cubique. Un voxel représente une valeur dans une grille régulière dans l'espace 3D. Une représentation à base de voxels peut être vue comme une discrétisation de la fonction définissant le volume (on considère dans ce cas que les points de l'espace n'appartenant pas à cette grille régulière finie ont une densité nulle).

Les représentations à base de voxels se reposent sur une représentation en grille régulière pour reconstruire des données volumiques. Toutefois, cette représentation n'est pas particulièrement adaptée au cas de données non-structurées. Une représentation plus efficace pour des données non-structurées est d'utiliser le point comme primitive de base. Le principe est d'échantillonner le volume par un nuage de points. De la même manière que pour toute représentation discrète, la reconstruction du volume passe par la définition en chaque point d'une fonction d'interpolation.

2.3 Représentation des propriétés microscopiques

L'apparence d'une surface ou d'un élément de volume est dépendant de l'interaction de sa microstructure avec les rayons lumineux. De plus, il est également nécessaire de prendre en compte l'influence de la millistructure pour des effets lumineux à plus grande échelle (par rapport à la microstructure). Ces niveaux de représentation sont souvent trop complexes pour se prêter directement à une modélisation. De plus, la complexité des phénomènes lumineux est trop importante pour permettre leur calcul exhaustif au niveau microscopique. Il est donc nécessaire de recourir à divers modèles permettant une approximation des interactions entre la lumière et un matériau. Nous commençons par discuter du modèle de réflectance bidirectionnelle, dans la section 2.3.1. Par la suite, nous présentons une extension de cette fonction prenant en compte les phénomènes de diffusion de la lumière à l'intérieur même d'un matériau (section 2.3.2), avant d'aborder les modèles couramment utilisés en informatique graphique (section 2.3.3).

2.3.1 Fonction bidirectionnelle de distribution de réflectance

Le comportement optique d'un matériau par rapport à une lumière incidente est décrit par sa fonction de réflectance f_r. Cette fonction décrit le rapport entre la luminance réfléchie et l'éclairement. L'éclairement dE d'un élément infinitésimal dS d'un matériau (non émissif) illuminé par une luminance incidente L_i arrivant selon une direction (θ_i, ϕ_i) au travers d'un angle solide infinitésimal $d\omega_i$ peut s'écrire :

$$dE(\theta_i, \phi_i) = L_i(\theta_i, \phi_i) \cos\theta_i d\omega_i \tag{2.3}$$

Le schéma 2.6 illustre ce principe.
La réflectance est une fonction *bidirectionnelle*, qui peut être exprimée comme le rapport entre la luminance quittant la surface L_r dans une direction (θ_r, ϕ_r) et de l'éclairement direct selon une direction ω_i [115] :

$$f_r(\theta_i, \phi_i, \theta_r, \phi_r) = \frac{L_r(\theta_r, \phi_r)}{dE(\theta_i, \phi_i)} = \frac{L_r(\theta_r, \phi_r)}{L_i(\theta_i, \phi_i) \cos\theta_i d\omega_i} \tag{2.4}$$

On emploie couramment le terme *BRDF* (de l'anglais : *Bidirectional Re ectance Distribution Function*) pour décrire cette fonction de réflectance. Cette fonction obéit à deux propriétés physiques fondamentales : la conservation de l'énergie et le principe de réciprocité de Helmhotz. Cette première propriété signifie que la somme des quantités d'énergie transmise, absorbée et réfléchie

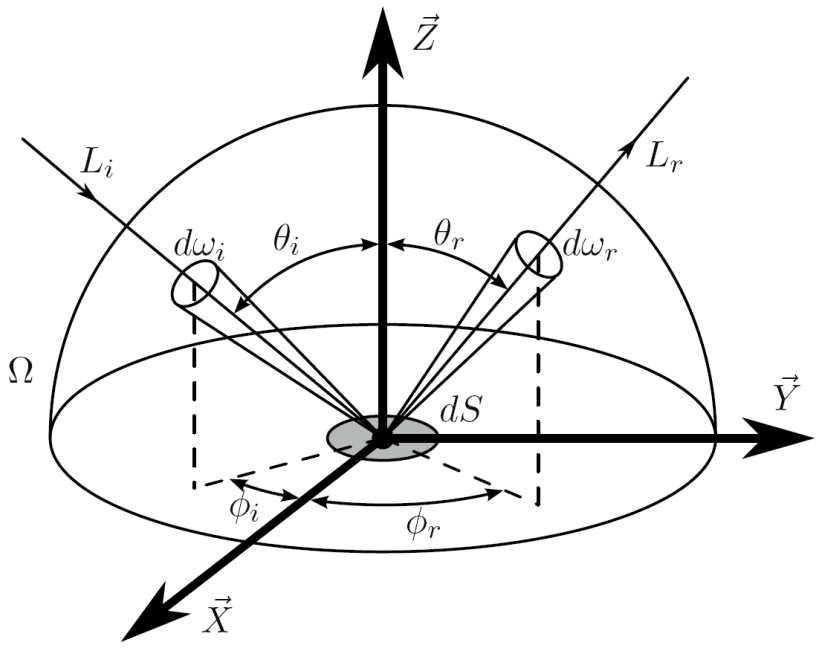


FIGURE 2.6 – La réflectance définit le flux d'énergie lumineuse L_r quittant le matériau dans la direction (θ_r, ϕ_r) en fonction de la direction de la luminance $L_i$incidente (θ_i, ϕ_i).

est égale à la totalité de l'énergie incidente. La deuxième loi physique établit que la valeur de la fonction de réflectance est identique si les directions d'incidence et de réflexion sont interverties. En règle générale, la BRDF dépend de beaucoup de paramètres au niveau microstructurel, comme par exemple l'état de la surface ou les indices de réfractions des éléments composant le matériau.

2.3.2 Diffusion de la lumière sous la surface

Le modèle des BRDF est toutefois basé sur une simplification courante et prend pour hypothèse que l'énergie lumineuse entre et sort d'un matériau au même point, ce qui n'est pas vrai pour tous les matériaux, comme par exemple les surfaces translucides. Le modèle des BSSRDF (de l'anglais *Bidirectionnal Subsurface Scattering Re ectance Distribution Function*) prend en compte la diffusion de la lumière à l'intérieur d'un matériau et peut être considéré comme une généralisation des BRDF. Une BSSRDF est une fonction exprimant la quantité d'énergie quittant le matériau au point x_o dans la direction $\vec{\omega_o}$ en fonction de la luminance entrant dans le matériau au point x_i avec la direction $\vec{\omega_i}$ et

peut s'écrire :

$$BSSRDF = S\left(x_i, \vec{\omega_i}, x_o, \vec{\omega_o}\right) \tag{2.5}$$

Avec ce modèle, il convient d'intégrer les contributions lumineuses sur une zone située autour du point x_o afin de connaitre la quantité de lumière émise en direction de $\vec{\omega_o}$, ce qui en pratique est difficilement réalisable pour des applications interactives. Jensen et al. [62] proposent toutefois le modèle du dipôle permettant une approximation du phénomène de diffusion de la lumière sous une surface.

2.3.3 Modèles de réflectance en synthèse d'image

Les fonctions de réflectance que nous venons de voir ont en pratique une dimensionnalité trop élevée pour être utilisées directement lors d'un rendu interactif. En pratique, un modèle plus simple à utiliser est appliqué pour reproduire les interactions lumineuses (en négligeant souvent le phénomène de diffusion sous la surface). En règle générale, les modèles de BRDF peuvent être regroupés en trois classes : les modèles physiques, explicites ou empiriques. Un modèle physique de BRDF se base sur un ensemble de statistiques sur les surfaces afin de prédire l'apparence finale du matériau. Un modèle explicite repose sur un échantillonnage d'une BRDF, pouvant être obtenu par mesure ou par simulation. Ce jeu de données permet de générer un tableau de valeurs (avec une fonction d'interpolation) qui peut être indexé lors de l'affichage. Les modèles empiriques se basent sur l'approximation empirique du comportement optique d'un matériau. Un modèle empirique largement utilisé dans le

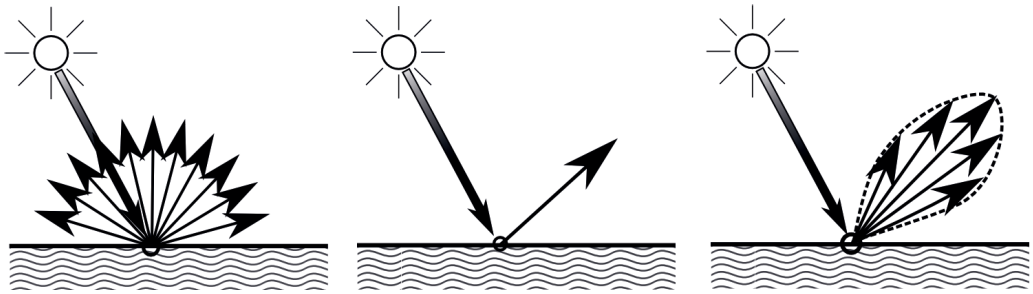


FIGURE 2.7 – Différents types de réflexions. A gauche : une réflexion diffuse sur une surface purement *Lambertienne*. Au milieu : une réfléxion unidirectionelle purement spéculaire (miroir parfait). A droite : un lobe spéculaire de réflexion autour de la direction de réflexion idéale (loi de Snell-Descartes)

domaine de l'informatique graphique est le modèle de Phong [125]. Ce mo-

dèle se base sur l'observation de deux types de réflexions : la réflexion diffuse et la réflexion spéculaire, comme présenté sur la figure 2.7. Le premier type de réflexion est la réflexion *lambertienne* dans laquelle la lumière incidente à un matériau est réfléchie de manière égale dans toutes les directions. Le deuxième type de réflexion constaté est la réflexion spéculaire. Ce type de réflexion concerne les miroirs parfaits et considère que la lumière est réflechie dans une seule direction de réflexion idéale par rapport à la loi de Snell-Descartes (tout rayon lumineux incident est réfléchi dans la direction opposée par rapport à la normale N de la surface du matériau). Cependant, les matériaux ne sont jamais parfaitement spéculaires. La structure de la microstructure engendre une dispersion effective de la lumière dans un cône autour de la direction idéale (à droite de la figure 2.7). Phong [125] propose de modéliser empiriquement la dispersion lumineuse sous la forme d'un lobe, en considérant la distance angulaire entre la direction de réflexion idéale et la direction d'observation du matériau. La modèle d'illumination de Phong, modélisant une surface ayant des propriétés à la fois lambertiennes et spéculaires, peut s'écrire :

$$f_{\text{Phong}}\left(\vec{v}, \vec{l}\right) = \rho_d \left\langle \vec{l} \circ \vec{n} \right\rangle + \rho_s \left\langle \vec{v} \circ \vec{l_{\text{idéal}}} \right\rangle^n \tag{2.6}$$

avec :

- $\vec{v}$ est le vecteur unitaire orienté vers l'observateur du matériau
- $\vec{l}$ est le vecteur unitaire orienté vers la source lumineuse
- $\vec{n}$ le vecteur normal à la surface du matériau au point considéré
- $\vec{l_{\text{idéal}}} = 2\left(\vec{l} \cdot \vec{n}\right)\vec{n} - \vec{l}$ est la direction de réflexion idéale
- ρ_d est le *coef cient diffus* du matériau, c'est-à-dire la propension du matériau à se comporter comme une surface lambertienne
- ρ_s est le *coef cient spéculaire* du matériau, c'est-à-dire la propension du matériau à se comporter comme un miroir
- n est le facteur modulant la forme du lobe (plus n est grand, plus le lobe est fin et plus la surface semble brillante)

Ce modèle est totalement empirique et repose essentiellement sur une définition correcte des paramètres ρ_s et ρ_d. Toutefois, bien que ce modèle soit éloigné d'une définition physique (par exemple, il ne respecte pas les principes de réciprocité de Helmholtz ou de conservation de l'énergie), il parvient néanmoins à retranscrire une illumination plausible. Bien que ce modèle ait été amélioré [81] pour respecter ces contraintes, la simplicité de l'évaluation du modèle de Phong contribue à en faire un des modèles d'illumination les plus populaires.

Il existe bien entendu de nombreux autres travaux [5] [32] [158] sur les modèles de représentation de BRDFs, mais nous nous sommes concentré ici sur le modèle le plus populaire et utilisé dans ces travaux.

2.4 Fonctions d'attribut

Afin de permettre une meilleure description des objets, il est souvent nécessaire de leur adjoindre diverses informations supplémentaires. En effet, un objet peut présenter des variations de couleur, de texture, ou de propriétés optiques. Il est important, afin de garantir une description efficace de l'objet, d'être capable également de représenter ses attributs. Dans ce but, il est courant d'ajouter au modèle une ou plusieurs fonctions d'attribut. Le but d'une telle fonction est de pouvoir définir la valeur d'un attribut (autre que la géométrie) en chaque point du modèle.

Cette fonction peut être définie de manière procédurale ou explicite. Une représentation procédurale est une représentation compacte se basant sur un jeu de paramètres. Elle permet d'effectuer le calcul de la valeur de l'attribut en chaque point du modèle lors de l'affichage de l'objet. Toutefois, il est courant de définir une telle fonction de manière explicite, notamment sous la forme d'une *texture*. Une texture est une discrétisation régulière finie de l'espace (généralement à une, deux ou trois dimensions) associée à un filtre simple (généralement linéaire) de reconstruction. Chaque élément de la texture (un *texel*, de l'anglais *texture element*) contient une valeur de l'attribut. Les textures présentent l'avantage d'être directement traitées par le matériel graphique.

Dans le cas de surface, l'application la plus connue est le *placage de texture* [24][18], qui consiste à plaquer une texture 2D (une image) sur une surface afin d'accroître sa richesse visuelle. Il faut pour cela mettre en correspondance chaque point de la surface 3D avec un point d'une image 2D en utilisant une paramétrisation. En pratique, cela consiste à associer à chaque sommet d'une surface explicite des coordonnées (u, v) correspondant à un point 2D de la texture, comme illustré par la figure 2.8. Ces *coordonnées de textures* sont des attributs interpolés linéairement entre les sommets (dans le cas de maillage polygonaux) afin de définir pour chaque point de la surface une correspondance avec un point de la texture 2D. Le problème de la définition de cette paramétrisation de surface est un problème non trivial et largement étudié qui dépasse néanmoins le cadre de cette thèse.

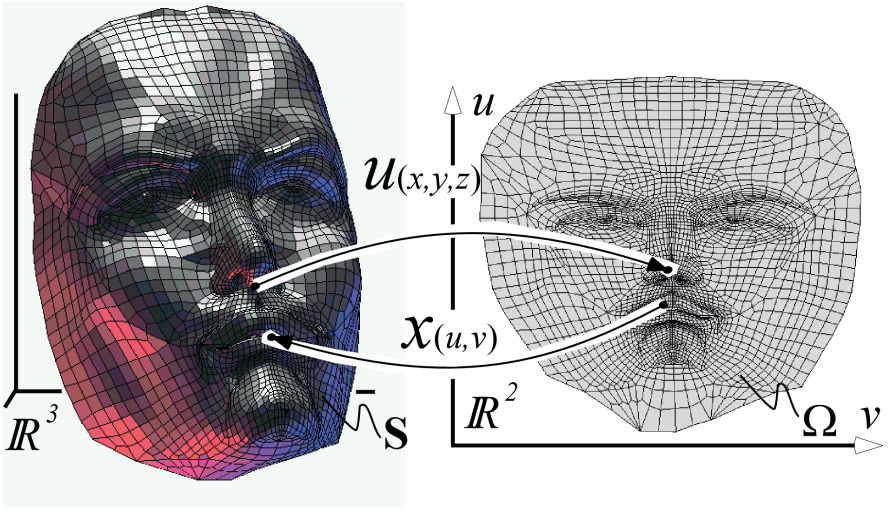


FIGURE 2.8 – Une paramétrisation $U()$ fait correspondre chaque point (x, y, z) d'une surface S de $\mathbb{R}^3$ à un point (u, v) appartenant à un sous-ensemble Ω de $\mathbb{R}^2$.Images de [99]

Conclusion

Nous avons vu dans ce chapitre un sous-ensemble des représentations habituellement utilisées dans le domaine de l'informatique graphique. Ces techniques sont axées sur la description de la géométrie des objets 3D et des propriétés microsurfaciques des matériaux. Ces représentations présentent chacune des avantages et des inconvénients selon les phénomènes que l'on souhaite représenter. Dans le domaine de la visualisation, ces représentations informatiques ont principalement vocation à être interprétées par une des nombreuses méthodes de rendu existantes, dont nous présentons les grands principes dans le chapitre suivant, afin de générer des images synthétiques.

Chapitre 3

Affichage interactif de scènes synthétiques

Nous avons décrit dans le chapitre précédent les principaux modèles permettant la description d'un objet dans un espace à trois dimensions. Dans ce chapitre, nous abordons les principes sous-jacents au *rendu*, c'est-à-dire à la génération d'images de synthèse. Le but de ce chapitre n'est pas la présentation exhaustive des techniques existantes, mais de se focaliser plutôt sur une présentation des grands principes du rendu. Nous mettons également l'accent sur les méthodes proches des travaux présentés dans cette thèse. Nous allons commencer par présenter le principe-même du rendu (section 3.1) avant de passer rapidement en revue le principe du pipeline graphique (section 3.2). Les méthodes de rendu peuvent être divisées selon les deux principales approches : les approches d'*ordre objet* et les approches d'*ordre image*. Les algorithmes d'ordre objet, présentés dans la section 3.3, itèrent sur les objets et calculent pour chaque objet sa contribution à l'image finale. Inversement, les algorithmes d'ordre image (section 3.4) considèrent un par un chaque pixel d'une image et s'attachent à calculer pour chaque pixel la contribution de l'ensemble de la scène. Ces deux grandes familles résolvent le problème commun de l'affichage 2D d'une scène synthétique mais possèdent des avantages et inconvénients très différents.

3.1 Principe du rendu

Le but du rendu en informatique graphique est l'affichage de scènes 3D sur un écran à deux dimensions. Nous nous focalisons dans cette thèse sur le rendu utilisant le *modèle de caméra à sténopé*. Ce modèle de caméra sans lentille est connu depuis l'Antiquité. Le principe découle de l'observation faite par le physicien Ibn al-Haytham [1] au $11^{\text{ème}}$ siècle que la vision résulte de rayons lumineux arrivant jusqu'à l'œil. Ce modèle consiste en une boîte fermée laissant passer les rayons lumineux par une petite ouverture dans un côté, permettant

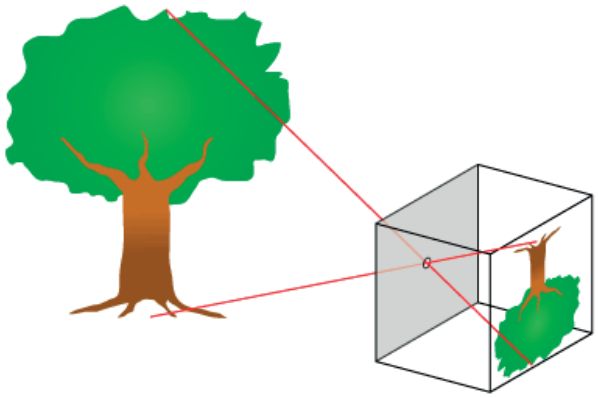


FIGURE 3.1 – Modèle de caméra à sténopé. La lumière se concentre en une petite ouverture projetant ainsi une image inversée.

la projection d'une image sur le côté opposé de la boîte (voir figure 3.1).
Ce modèle, largement utilisé, a donné lieu à un modèle opposé, présenté sur la figure 3.2, où le plan de projection de l'image se trouve devant le centre optique (le point de convergence des rayons lumineux) plutôt que derrière.

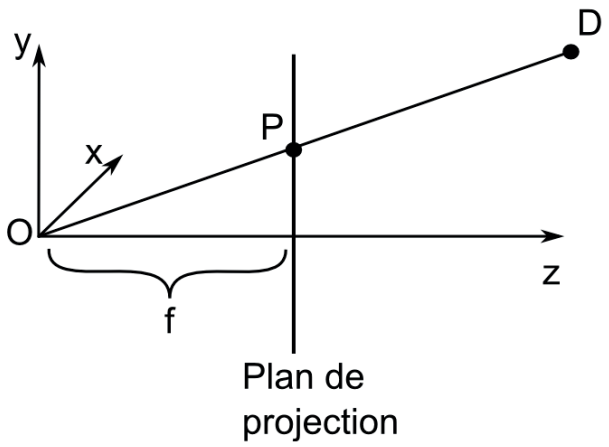


FIGURE 3.2 – Dans ce modèle de caméra, le plan de projection se trouvant devant le centre optique O.

Dans ce modèle, la projection d'un point D de l'espace 3D de la scène sur le plan 2D de projection de l'image peut s'écrire comme un produit matrice-vecteur en coordonnées homogènes en considérant le centre optique du modèle O comme l'origine de l'espace 3D : $P = M_{\text{Proj}} D$ avec M_{Proj} définie par :

$$M_{Proj} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.1}$$

avec f la distance focale (distance entre le centre optique et le plan de projection).

En pratique, le plan image sur lequel l'image est projetée est discrétisé et affiché sur un écran standard. Afin de simplifier et de nous affranchir des problèmes de résolution, nous allons par la suite considérer que la discrétisation de notre plan image correspond à la résolution de notre périphérique d'affichage. Ceci nous permet d'établir la correspondance : une division du plan ↔ un pixel (de l'anglais *picture element*) de l'image finale. Au final, le principe du rendu est dans ce cas d'attribuer une couleur à chaque pixel, afin de reconstituer le cheminement de la lumière depuis une source lumineuse jusqu'à l'œil. Nous allons commencer par distinguer deux cas, le cas des scènes définies de manière surfacique (section 3.1.1) et les scènes définies de manière volumique (section 3.1.2). Nous prenons ici volontairement et par souci de simplicité les cas extrêmes où l'ensemble de la scène est défini soit de manière totalement surfacique, soit de manière totalement volumique par une information de densité en chaque point. Cependant, il est bien entendu possible de représenter des scènes contenant les deux types d'éléments.

3.1.1 Rendu surfacique

L'hypothèse majeure dans un rendu totalement surfacique est la non-interaction de l'espace situé entre les surfaces définissant les objets et un rayon lumineux. C'est-à-dire que l'on considère dans ce cas que l'espace entre les objets est composé uniquement de vide et n'influe pas sur la trajectoire d'un rayon de vue. Dans ce cas, le principe d'un rendu consiste en premier lieu à déterminer pour chaque pixel quelles sont les surfaces ayant reflété, absorbé ou été traversées par le rayon lumineux arrivant en ce pixel. Afin de simplifier, nous allons considérer que ce problème s'apparente à un problème de visibilité, c'est à dire consiste à déterminer pour chaque pixel quelle est la dernière surface à avoir interagi avec le rayon lumineux. Le problème s'apparentera ensuite à estimer quelle était la composition, direction et contribution du rayon lumineux qui, arrivant à cette surface a été réfléchi/transmis/propagé en direction du pixel considéré.
Pour ce faire, Kajyia [64] propose d'intégrer l'ensemble des contributions des sources lumineuses en pondérant chaque contribution par la BRDF de la surface au point considéré. Cette équation est l'*équation d illumination globale* et

exprime la quantité de lumière refléchie en direction de $\vec{\omega_o}$ pour tous les points de toutes les surfaces de la scène :

$$L_o(\vec{\omega_o}) = L_e(\vec{\omega_o}) + \int_\Omega \text{BRDF}(\vec{\omega_i}, \vec{\omega_o}) L_i(\vec{\omega_i}) \cos(\theta_i) \text{d}\, \vec{\omega_i} \tag{3.2}$$

avec θ_i l'angle entre la normale à la surface et la direction incidente $\vec{\omega_i}$ (voir figure 2.6). Cette équation exprime la quantité de lumière réfléchie en direction de $\vec{\omega_o}$ en intégrant sur la sphère des directions (Ω) l'éclairement direct pondéré par la BRDF du matériau. $L_e\ \vec{\omega_o}$ représente la luminance propre du point de la surface émise en direction de $\vec{\omega_o}$. Dans le cas (fréquent) de sources lumineuses ponctuelles, cette intégrale s'approxime en pratique par la somme des contributions de chaque source lumineuse.

3.1.2 Rendu volumique

La visualisation directe [43][67][152] de données volumiques pose un problème de conception. En effet, le système visuel humain étant principalement habitué à percevoir des surfaces, l'idée de représenter des données purement volumiques est peu intuitive. De plus, une telle représentation, de par la masse de données à représenter, est lourde et coûteuse (en terme de calculs). Les premières méthodes de visualisation parvenaient à un affichage en se basant sur une conversion du modèle volumique en modèle surfacique (par exemple en procédant à l'extraction de l'interface entre deux structures volumiques de densité différentes).

Avant de s'attaquer au problème du rendu volumique, il convient de définir les informations que l'on veut visualiser. En règle générale, lorsque l'on s'attache à visualiser des données volumiques, on aimerait pouvoir distinguer clairement les données tridimensionnelles, les structures internes et leurs relations spatiales. Un modèle intuitif permettant de visualiser ces informations consiste en l'affichage des données volumiques sous la forme de structures ou primitives semi-transparentes. Ce modèle intuitif est largement admis par la communauté graphique et permet la visualisation directe de données volumiques.

Théorie du rendu volumique

Ce qu'on entend habituellement par « rendu volumique direct » est une approximation de la propagation de la lumière à travers un médium participatif

(le volume), composé de particules avec des propriétés d'absorption et d'émission de lumière. Alors que la lumière traverse le volume, elle interagit avec ces particules et peut être absorbée, réfléchie, ou émise. Les équations du transport de la lumière ont été utilisées pour simuler la transmission de la lumière à travers un volume par Kajiya [67] et Tuy et al. [152]. De manière similaire, Blinn [17] utilise une approximation de cette équation pour modéliser les fonctions de réflectance de volumes définis par un ensemble de couches.

Le but du rendu volumique étant la visualisation, le principe est de calculer quelle quantité de lumière traversant le volume arrive sur chaque pixel du plan image. L'interaction de la lumière avec les éléments de volume est décrite par la théorie du transport de la lumière [64, 75, 137], et permet de dériver l'équation du transfert pour un milieu volumique. L'équation utilisée en rendu volumique direct est une approximation construite en négligeant les effets de dispersion de la lumière (la lumière ne dévie pas de son trajet rectiligne) :

$$I(D) = I_0 e^{-\int_{s0}^{D} \kappa(t)dt} + \int_{s0}^{D} q(s) e^{-\int_{s}^{D} \kappa(t)dt} ds \tag{3.3}$$

Dans cette équation, $q(s)$ peut être interprété comme un coefficient d'émission d'une particule situé à une distance s de la caméra le long du rayon de vue. κ représente de la même manière le coefficient d'absorption d'une particule.
Cette équation exprime la luminance $I(D)$ quittant le volume au point $s = D$ et atteignant la caméra pour une luminance I_0 entrant dans le volume au point $s = s0$ (voir figure 3.3).

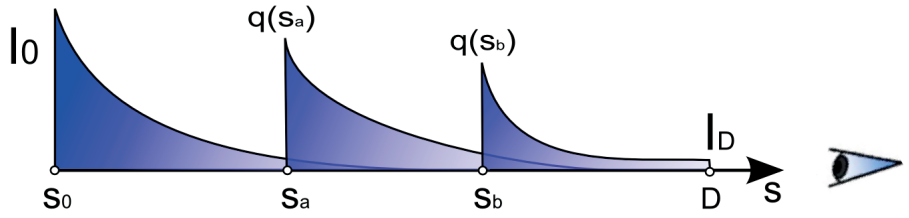


FIGURE 3.3 – L'équation du rendu volumique permet de calculer la quantité de lumière $I(D)$ arrivant jusqu'à l'oeil en fonction de l'intensité entrant dans le volume $I(s_0)$ en prenant en compte l'émission d'énergie $q(s)$ ainsi que le phénomène l'absorption d'énergie des particules (s_a et s_b)situées le long du rayon.

Le premier terme de cette équation décrit la lumière entrant dans le volume et atténuée par les particules du volume. Le second terme représente quant à lui les contributions des particules atténuées par le média participatif avant

d'atteindre la caméra. On peut définir :

$$\tau(s_a, s_b) = \int_{s_a}^{s_b} \kappa(t)\,dt \tag{3.4}$$

comme étant la profondeur optique entre les positions s_a et s_b. Ce terme peut s'interpréter comme la distance de propagation de la lumière avant qu'elle ne soit diffusée ou absorbée par le médium. Pour simplifier, ce terme permet de représenter la transparence d'une particule, c'est-à-dire qu'il indique si la particule transfère (ou absorbe, dévie) un rayon lumineux incident. L'information de transparence (pour un volume entre s_a et s_b) peut s'écrire :

$$T(s_a, s_b) = e^{\tau(s_a, s_b)} = e^{\int_{s_a}^{s_b} \kappa(t)\,dt} \tag{3.5}$$

Afin de résoudre cette équation, des méthodes numériques doivent être employées. Une approche courante est de procéder à une division du domaine d'intégration en n échantillons successifs. La radiance au point s_i peut ainsi s'exprimer :

$$I(s_i) = I(s_{i-1})\,T(s_{i-1}, s_i) + \int_{s_{i-1}}^{s_i} q(s)\,T(s, s_i)\,ds \tag{3.6}$$

On peut réécrire la transparence et la couleur du $i^{\text{ème}}$ intervalle par :

$$\begin{aligned} T_i &= T(s_{i-1}, s_i) \\ c_i &= \int_{s_{i-1}}^{s_i} q(s)\,T(s, s_i)\,ds \end{aligned} \tag{3.7}$$

Partant de cette notation simplifiée, il est possible d'écrire la quantité de lumière arrivant à la caméra comme :

$$I(D) = I(s_n) = I(s_{n-1})\,T_n + c_n = (I(s_{n-2})\,T_{n-1} + c_{n-1})\,T_n + c_n = \ldots \tag{3.8}$$

Ceci peut s'écrire de manière plus compacte :

$$I(D) = \sum_{i=0}^{n} c_i \prod_{j=i+1}^{n} T_j, \text{ avec } c_0 = I(s_0). \tag{3.9}$$

En régle générale, la transparence T_i est remplacée par une valeur d'opacité $\alpha_i = 1 - T_i$.
L'équation 3.9 nous donne une base pour une résolution itérative de l'intégrale du rendu volumique. Une méthode de résolution est l'application successive d'opérations le long d'un rayon de vue. Lorsque l'on choisit un schéma de composition de l'avant vers l'arrière (*front to back*) par exemple, l'idée sera d'accumuler les contributions des éléments de volumes (i) situés aux échantillons du

rayon de vue, en parcourant le rayon de vue depuis la caméra (l'avant) jusqu'à l'entrée du rayon dans le volume (l'arrière) :

$$\begin{aligned} C_{acc} &= C_{acc} + (1 - \alpha_{acc}) C_i \\ \alpha_{acc} &= \alpha_{acc} + (1 - \alpha_{acc}) \alpha_i \end{aligned} \tag{3.10}$$

avec C_i et α_i les coefficients de couleur et d'opacité du volume à la position de l'échantillon i du rayon de vue. Il est à noter ici que C_i représente un coefficient de *couleur associée*, tel qu'introduit par Blinn [15]. Cela signifie simplement que le coefficient C_i correspond à la couleur pondérée par l'opacité correspondante.

Fonction de transfert de couleur et d'opacité

Nous avons pu voir à l'instant que l'équation du rendu volumique se base sur les propriétés de couleur et d'opacité des éléments du volume. Les volumes étant majoritairement modélisés par une information de densité, les propriétés optiques des particules doivent être déduites de cette densité. Ceci est réalisé en s'aidant d'une *fonction de transfert*. Cette fonction de transfert établit simplement une correspondance entre une information de densité et des coefficients de couleur c et d'opacité α. Cette fonction de transfert est critique pour déterminer la manière dont les données seront représentées dans le rendu final. Cette conversion d'une valeur de densité en valeurs d'opacité et de couleur par le biais d'une fonction de transfert est appelée étape de *classi cation* dans la communauté de visualisation scientifique. Les coefficients de valeur et d'opacité des échantillons permettent de calculer l'intégrale 3.7 pour un segment du rayon de vue. Cette intégrale peut être approximée par discrétisation le long du segment. L'étape de classification peut être réalisée avant l'application du filtre de reconstruction (pré-classification) ou après (post-classification). Le schéma de pré-classification revient à assigner une couleur et une opacité aux extrémités S_f et S_b d'un segment le long du rayon de vue et à interpoler ces valeurs pour chaque échantillon le long du segment (S_f, S_b). Au contraire, le schéma de post-classification consiste à interpoler les valeurs scalaires de densité le long du segment et à assigner les propriétés optiques à chaque échantillon en indexant la fonction de transfert avec la valeur interpolée (voir la figure 3.4).

Ces deux schémas peuvent poser des problèmes d'échantillonnage, selon la définition de la fonction de transfert. En effet, dans le cas d'une fonction de transfert contenant un signal à haute fréquence, il est nécessaire d'augmenter

significativement le nombre d'échantillons utilisés pour approximer l'intégrale 3.3 sans artefacts [73]. Une idée courante pour réduire ce problème est celle de la pré-intégration [46, 134]. Basiquement, l'idée est de précalculer l'intégrale de l'équation 3.7 définissant les coefficients de couleur et d'opacité le long d'un segment du rayon de vue. Ce précalcul s'effectue à partir des densités aux extrémités du segment en utilisant un certain nombre d'approximations, comme la linéarité de la variation de l'intensité le long du segment. En pratique, cette technique produit une table à trois dimensions indexée par les densités aux extrémités du segment et par la longueur du segment. Cette table à trois dimensions est souvent approximée par une table à deux dimensions en considérant une longueur de segment constante (ce cas est valable lorsque l'échantillonnage le long d'un rayon de vue est réalisé avec un pas constant) (voir figure 3.4).

3.1.3 Aliassage

Le phénomène connu d'*aliassage* (*aliasing* en anglais) est directement lié à la discrétisation du plan image et est la cause d'artefacts visuels. Ce phénomène est causé par la représentation d'un signal à haute fréquence (la description d'une scène par exemple) par un signal discret à résolution insuffisante (le plan image discrétisé). Un des artefacts visuels le plus courant est l'effet de crénelage, présenté sur la figure 3.5. Un pixel du plan image est associé à une seule et unique couleur, représentant les objets ou interactions lumineuses visibles dans le cône engendré par le centre optique O et le pixel P. Par exemple, sur la figure 3.6, plusieurs objets contribuent à la couleur du pixel P. Afin de rester cohérent, la couleur finale de P doit donc être un mélange des contributions de ces objets. La plupart des méthodes interactives se contentent cependant d'une liste finie de contributions et assignent la couleur résultant d'une unique contribution au pixel. Ce phénomène crée des artefacts variés tels les effets d'escaliers ou des erreurs de visibilité.

Une technique commune à de nombreuses méthodes de rendu et permettant la diminution de ces artefacts consiste à augmenter l'échantillonnage du plan image. En effet, un échantillonnage plus élevé permet la représentation d'un contenu hautes fréquences sans artefacts. Toutefois, la résolution utilisable par les périphériques physiques (les écrans actuels par exemple) reste limitée. Une des méthodes les plus répandue dans l'industrie consiste à suréchantillonner massivement le plan image lors du rendu et à réduire la résolution du plan image (en moyennant les contributions des pixels) avant affichage sur un périphérique physique, comme illustré par la figure 3.7. Cette technique

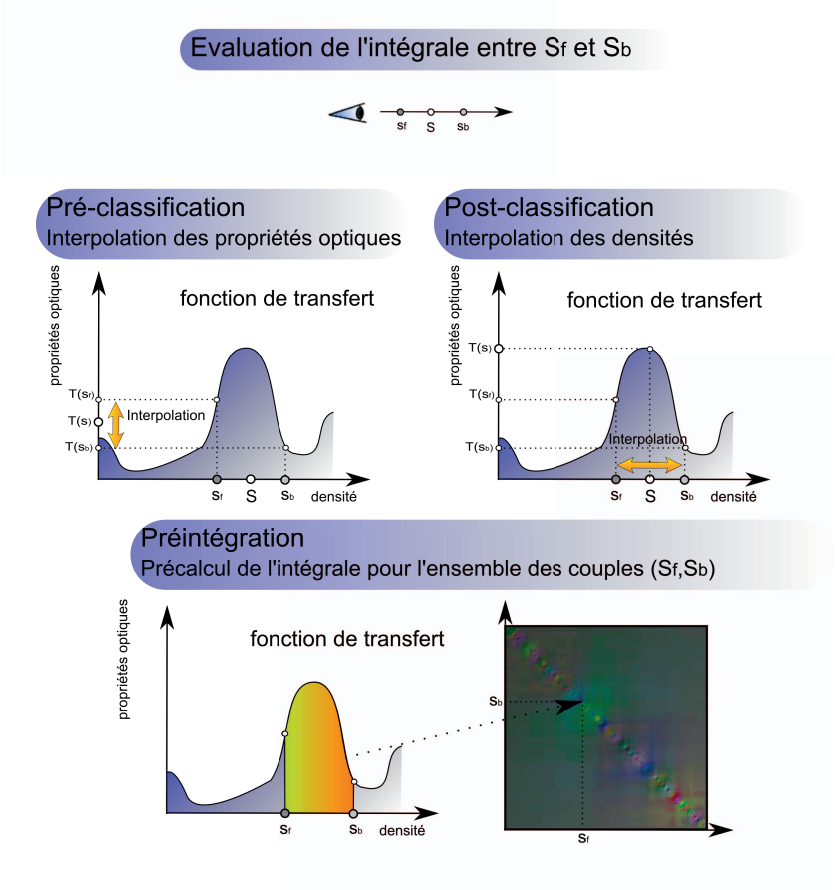


FIGURE 3.4 – Afin de résoudre l'intégrale 3.7 entre S_f et S_b, il est possible d'échantillonner le rayon de vue. Afin de déterminer la valeur d'un échantillon S, il est possible d'utiliser la fonction de transfert avant le schéma de reconstruction (pré-classification) ou après (post-classification). Les deux schémas peuvent causer une erreur d'interpolation lorsque peu d'échantillons sont utilisés. La technique de préintégration permet de résoudre ces erreurs d'interpolation en précalculant l'intégrale pour tous les couples (S_f, S_b).

permet de simuler les contributions de plusieurs rayons lumineux et permet une réduction des artefacts d'aliassage, au prix toutefois de performances réduites dues à la manipulation du plan image à résolution accrue. Cependant, il est à noter que cette méthode n'élimine pas complètement les artefacts d'aliassage mais ne fait que repousser la limite à laquelle ce phénomène intervient.

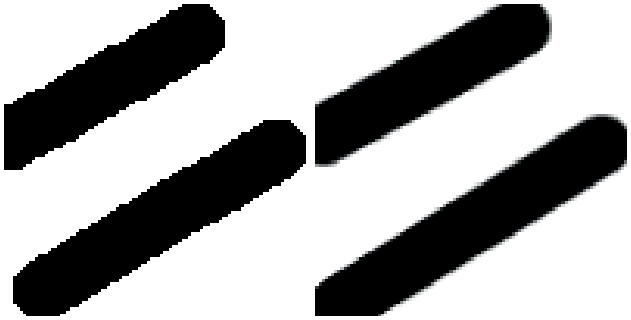


FIGURE 3.5 – À gauche, des lignes avec un artefact de crénelage. A droite, la même image avec un effet minimisé (anti-aliasing).

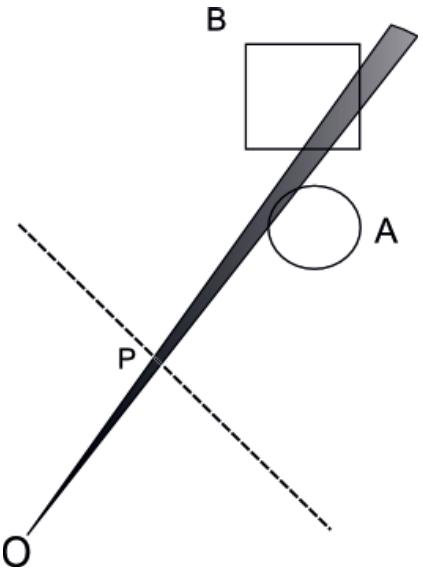


FIGURE 3.6 – Les objets *A* et *B* doivent contribuer à la couleur finale du pixel *P*.

3.2 Pipeline du processeur graphique

De nos jours, le rendu interactif de scènes synthétiques est étroitement lié à l'utilisation du matériel graphique. Ce dernier a connu une évolution importante au cours de la dernière décade. La puissance du GPU (de l'anglais *Graphics Processing Unit*, processeur dédié au rendu 3D) évolue de manière plus rapide que la loi de Moore pour les processeurs centraux[1]. De nombreuses

1. La loi de Moore estime (empiriquement) que la densité des transistors sur une puce double tous les 18 mois.

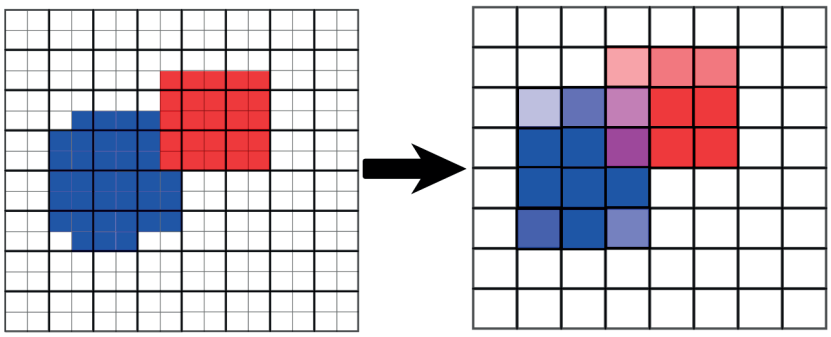


FIGURE 3.7 – Une technique classique d'anti-aliasing consiste à suréchantillonner le plan image lors du rendu et à mélanger les contributions des pixels avant affichage avec une résolution plus faible.

fonctionnalités s'ajoutent régulièrement, contribuant à rendre l'utilisation d'une carte graphique indispensable de nos jours pour un rendu interactif. Le GPU opère grâce à un pipeline s'occupant du traitement des primitives (principalement explicites) composant les objets et de la production d'une image finale, grâce à une implémentation matérielle des techniques de rendu. Le GPU est fortement orienté vers le rendu rapide de modèles polygonaux (principalement des triangles) de par son implémentation matérielle efficace des algorithmes de rasterisation. De plus, le GPU propose un traitement rapide des textures en tirant parti d'une implémentation matérielle d'une interpolation trilinéaire (voir section 2.2). Les capacités de programmation des différentes étapes du pipeline graphique offrent aujourd'hui une souplesse dans l'utilisation du GPU et permettent d'utiliser pleinement ses capacités pour offrir un rendu interactif de phénomènes complexes.

Lors du rendu d'une scène synthétique en utilisant le GPU, l'application ne calcule plus directement l'apparence mais envoie les informations relatives à cette dernière au matériel graphique, qui prend en charge les diverses étapes composant le pipeline graphique. Ces informations consistent en une description de la scène (souvent stockée directement en mémoire graphique lors de l'initialisation) et des informations relatives à la caméra, aux lumières ou aux textures.
La figure 3.8 illustre les principaux composants du pipeline graphique. Son fonctionnement repose sur la transformation successive des primitives par chacune des composantes. La puissance des cartes graphiques repose principalement sur le caractère parallélisable de ce pipeline. En effet, chaque primitive traverse le pipeline indépendamment des autres. Chaque primitive passe en

premier lieu par l'étape de transformation, qui consiste à projeter la primitive 3D dans l'espace image. L'étape de « rasterisation » ou tramage génère à partir de la projection de chaque primitive un ensemble de *fragments*, auxquels l'étape suivante associe une couleur. Ces fragments sont une discrétisation de la projection d'une primitive dans l'espace image et peuvent être considérés comme des « pixels intermédiaires » associés à une primitive, c'est-à-dire des éléments à une position fixe dans l'espace image qui contribueront à définir la couleur du pixel final. Les attributs associés à chaque sommet sont interpolés et transmis à chaque fragment. Chaque fragment dispose également d'une information de profondeur relative à la distance de la primitive à la caméra. Nous appellerons par la suite fragment tout élément généré par cette étape de rasterisation.

Plusieurs de ces étapes sont désormais programmables, ce qui permet de moduler leur comportement :

Transformation et *Vertex program* Ce programme est effectué une fois pour chaque sommet d'une primitive et permet de modifier l'étape de transformation du pipeline, c'est-à-dire la manière dont la primitive est projetée sur le plan image. Ce programme doit veiller à générer pour le sommet de la primitive les coordonnées correspondant à sa projection 2D. Il est également possible au cours de cette étape d'effectuer tous les calculs associés aux attributs du sommet (par exemple calcul de la normale, de la direction de l'illumination, des coordonnées de textures,....).

Geometry shader Ce programme, disponible sur les cartes graphiques, à partir de la série des GeForce 8 de NVIDIA, permet de générer des primitives supplémentaires directement dans le plan image. Il est donc possible dans cette étape de créer un ensemble de primitives pour chaque primitive entrant dans le pipeline.

Ombrage et *Fragment program* Le cœur de cette étape consiste à associer à chaque fragment une couleur. Ce programme est appelé une fois pour chaque fragment généré au cours de l'étape de rasterisation. Il est possible d'effectuer de nombreux calculs durant cette étape, comme par exemple, les calculs d'illumination ou les opérations de texture. Chaque fragment a une position fixe (qu'il n'est pas possible de modifier) et reste indépendant des fragments voisins.

Cet ensemble de programmes permet aujourd'hui de réaliser rapidement un nombre important d'effets et de calculs auparavant difficiles à effectuer en temps réel. Il subsiste toutefois un certain nombre de contraintes liées à l'architecture. Néanmoins, les performances actuelles du matériel graphique ouvrent également la porte à une utilisation du GPU dépassant le cadre de l'informa-

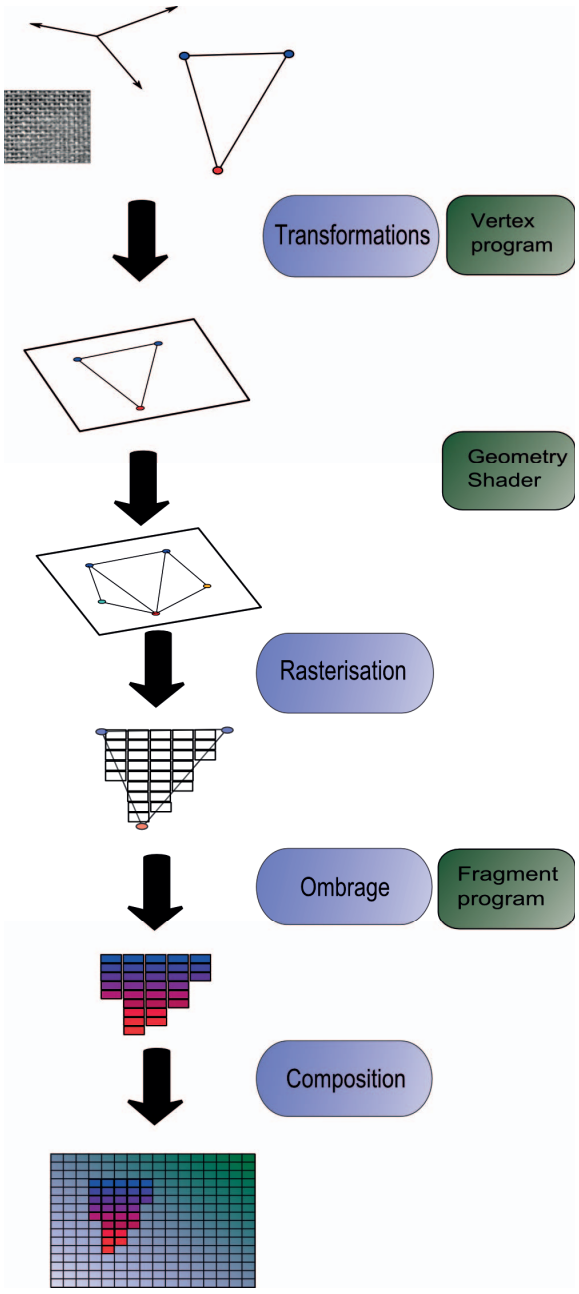


FIGURE 3.8 – Étapes principales du pipeline graphique, de la description d'une scène à l'affichage de l'image finale.

tique graphique. Une voie importante de recherche, le GPGPU (de l'anglais *General Purpose GPU*), se dédie à utiliser la puissance du GPU pour des calculs divers, incluant par exemple l'algèbre linéaire [78] ou divers solveurs [56]. L'uti-

lisation du GPU à des fins autres que le rendu d'une scène est désormais portée par des environnements dédiés tels *CUDA* ou *OpenCL*.

3.3 Algorithmes d'ordre objet

Les algorithmes d'ordre objet composent une image en considérant la projection des objets sur le plan image. Le principe est de résoudre le problème de visibilité en effectuant la projection de chaque primitive de la scène sur le plan image, générant ainsi un ensemble de fragments. Lorsque plusieurs primitives se projettent sur le même pixel (c'est-à-dire plusieurs fragments pour un même pixel), un mécanisme (matériel) de tampon de profondeur (*Z-Buffer*) permet de sélectionner pour chaque pixel le fragment qui sera retenu pour l'image finale. Il s'agit généralement du fragment ayant la profondeur la plus petite, c'est-à-dire le fragment issu de la primitive la plus proche de la caméra.

Cet algorithme repose lourdement sur le principe de la *rasterisation* (tramage en français). Ce mécanisme permet principalement de reconstruire l'image de la projection d'une surface ou d'un volume dans le cas d'objets définis de manière explicite. En effet, ces primitives étant souvent représentées par des points dans l'espace, leur projection sur le plan image donne également des points dans le plan image. Afin de reconstruire l'objet en lui-même, l'étape de rasterisation consiste à « remplir » l'espace contenu entre ces points afin de reconstruire la projection de la primitive. La figure 3.9 illustre ce mécanisme pour le cas de surfaces définies par des triangles. Il est à noter que ce mécanisme permet en outre l'interpolation linéaire (entre les sommets) des diverses informations attachées à chaque sommet.
Nous allons voir dans cette section différentes méthodes permettant le rendu d'objets définis de manière surfacique (section 3.3.1) ou volumique (section 3.3.2).

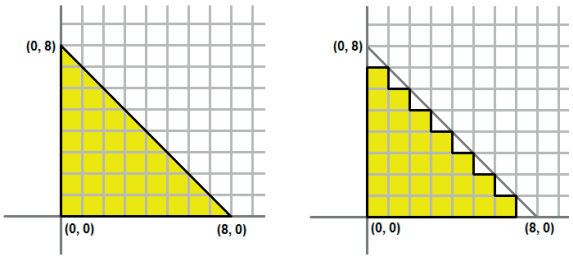


FIGURE 3.9 – La rasterisation permet de déterminer quels sont les fragments appartenant à la projection d'un triangle sur le plan image.

3.3.1 Surfacique

Dans le cas de surfaces définies par un ensemble de polygones, la méthode de projection/rasterisation est utilisée de manière majoritaire. L'utilisation du matériel graphique permet un rendu rapide des maillages polygonaux, en utilisant le pipeline présenté sur la figure 3.8. La méthode se résume à la projection des sommets des primitives sur le plan image et à la génération d'un ensemble de fragments (correspondant à la surface) au cours de la rasterisation. Chaque fragment possède généralement des paramètres (interpolés depuis les paramètres associés aux sommets) qui permettent l'évaluation des interactions entre la lumière et le matériau ou l'indexation d'une texture afin d'obtenir une couleur spécifique.

En ce qui concerne les représentations à base de *surfels* (voir section 2.1.3), les capacités de programmation des cartes graphiques permettent la reconstruction de l'élément de surface en utilisant un *fragment program*. Ces méthodes dites de *rendu par points* (*splatting*) [90, 136, 176, 177] connaissent un succès de par leur efficacité et leur fidélité de reconstruction. L'idée est de rasteriser chaque primitive par un carré englobant l'empreinte de la projection de la primitive (une ellipse, par exemple) et de déterminer à l'aide du *fragment program* quels sont les fragments appartenant réellement à l'élément de surface. Les fragments concurrents (des contributions pour le même pixel) sont mélangés si leur profondeur est voisine (ils sont issus de surfels voisins contribuant au même point de la surface 3D). Ceci permet d'obtenir un rendu de surface lisse et fidèle à la représentation originale (voir figure 3.10).

La méthode de projection/rasterisation est toutefois totalement inadaptée aux primitives définies de manière paramétrique ou implicite. La principale al-

FIGURE 3.10 – Le rendu par points permet une représentation lisse de la surface d'origine. Ici, une image rendue avec la technique du *phong splatting* [19].

ternative de rendu pour ces primitives consiste souvent en l'extraction d'une géometrie polygonale [52, 98] (qui peut alors être rendue par rasterisation). Il est également possible d'échantillonner la surface par un jeu de points et d'en effectuer le rendu sur le GPU [28] ou en utilisant un système hybride à base d'isosurface [95].

3.3.2 Volumique

Le principe des méthodes de rendu volumique de cette section est d'évaluer l'équation du rendu volumique (équation 3.3) en utilisant une méthode basée sur une projection des éléments sur le plan image. Il existe plusieurs méthodes, présentant chacune différents avantages et inconvénients et plus ou moins adaptée aux différentes représentations des données. Un des problèmes majeurs auquel on se heurte lors de l'affichage d'un objet volumique par rasterisation en utilisant le matériel graphique est l'absence de primitives volumiques au sens strict du terme. La solution à ce problème consiste à recourir à une *géométrie de substitution* (*proxy geometry*). En pratique, cette géométrie de substitution consiste en un ensemble de polygones ou de points englobant ou échantillonnant le volume qui « se substitue » aux primitives volumiques. Cette géométrie (surfacique) peut être rasterisée et permet de générer un ensemble de fragments auxquels on associera une couleur en fonction des données volumiques.

Méthode de slicing

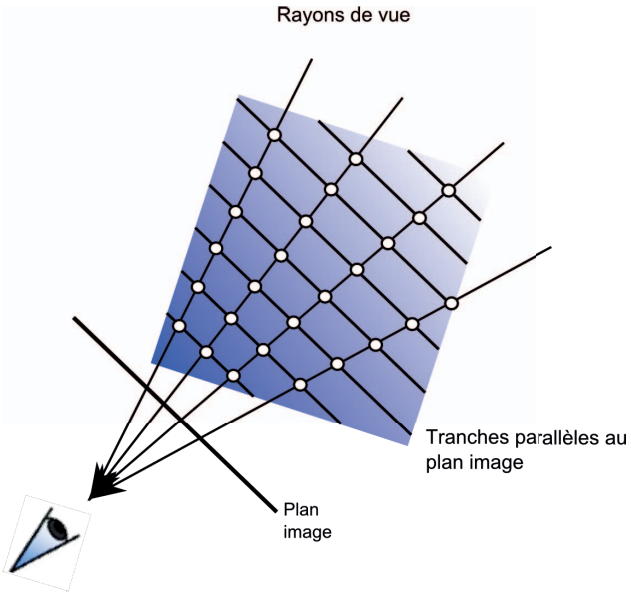


FIGURE 3.11 – Avec une méthode de *slicing*, le volume est découpé en tranches (ici parallèles au plan image). Les tranches successives sont projetées sur le plan image et composées. Il en résulte un échantillonnage inconsistant pour chaque rayon de vue.

La première approche [33, 21] que nous présentons est largement utilisée

et se base sur un ensemble de polygones (définis dans le repère d'un cube unité) parallèles au plan image (des « tranches ») comme géométrie de substitution afin de recréer l'échantillonnage d'un rayon de vue traversant le volume, comme illustré en 2D par la figure 3.11. À chacun des sommets de ces polygones est associée une coordonnée dans l'espace 3D correspondant à un cube unité. Ces coordonnées peuvent directement être utilisées pour indexer la texture 3D contenant les données volumiques (l'espace de définition du cube unité étant identique à l'espace de définition des coordonnées de la texture).

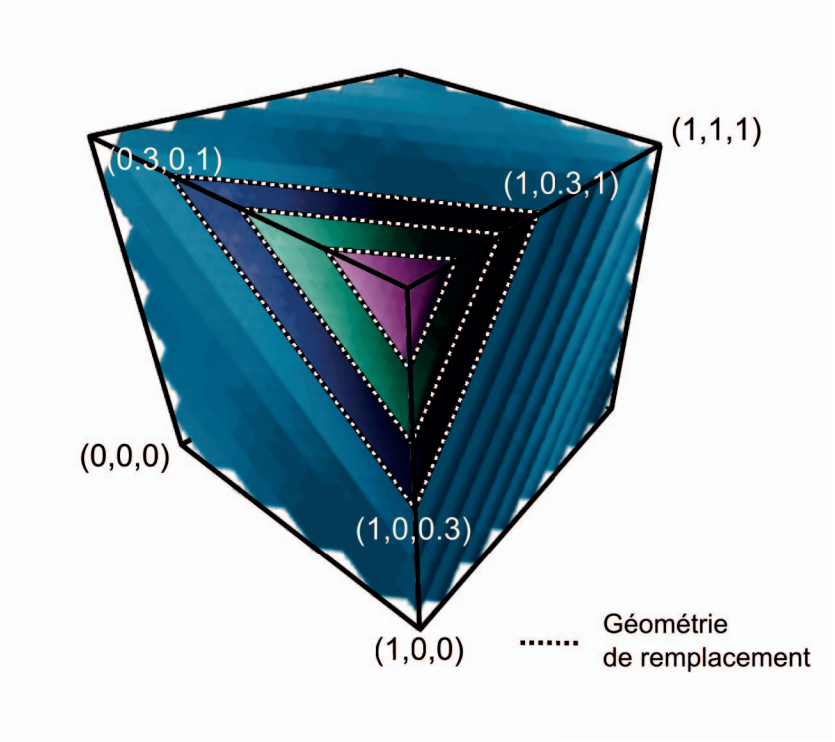


FIGURE 3.12 – Principe des méthodes de *slicing* avec une texture 3D. Une géométrie de substitution est créée sous la forme d'un ensemble de polygones intersectant le volume et parallèles au plan image. Ils sont projetés sur le plan image, créant un ensemble de fragments. La composition des contributions des fragments recompose l'image finale.

Le principe, illustré par la figure 3.12, est le suivant :

- Création des tranches parallèles au plan image intersectant le cube unité (matérialisant un cube englobant la texture 3D).
- Calcul des coordonnées de chaque sommet des polygones (coordonnées qui permettront d'indexer la texture 3D).

– Projection sur le plan image des polygones (ordonnés), créant ainsi un ensemble de fragments.
– Chaque fragment indexe la texture 3D (en fonction de ses coordonnées), en déduit des coefficients de couleur et d'opacité et accumule le résultat dans l'image finale d'après l'équation 3.9.

Cette méthode, largement répandue, tire pleinement parti de l'implémentation matérielle de l'interpolation trilinéaire du matériel graphique. Toutefois, l'utilisation de cette méthode implique d'avoir des données pouvant être stockées sous la forme d'une grille régulière. Il est également important de considérer le nombre de polygones (de tranches dans la texture) qu'il est nécessaire de générer pour représenter les données de manière fidèle . En effet, cette géometrie de substitution influe directement sur le taux d'échantillonnage de chaque rayon de vue et doit être suffisant afin de ne pas introduire d'artefacts. De plus, le taux d'échantillonnage n'est pas consistant entre les différents rayons de vue. Ce phénomène conduit à des artefacts dus à l'augmentation du taux d'échantillonnage sur les bords de l'image (voir figure 3.11). Ceci est principalement dû à l'utilisation d'une géométrie de substitution planaire. Il est toutefois possible de se débarrasser de ces artefacts en utilisant des primitives sphériques, comme proposé par LaMar et al. [83], au prix toutefois de performances amoindries : le coût de création et de projection de primitives sphériques est beaucoup plus important qu'une projection de primitives planaires.

Méthode de splatting

Les méthodes dites de *splatting*, introduites par Westover [163] sont naturellement adaptées au rendu de données volumiques explicites définies par un ensemble de points. Basiquement, cette méthode représente le volume par un ensemble de fonctions de reconstruction (voir section 2.2). L'image finale est générée par la projection de ces noyaux de reconstruction G_i dans le plan image. Pour ce faire, chaque échantillon du volume est projeté dans l'espace écran. L'idée est d'utiliser la rasterisation pour générer un ensemble de fragments qui permettront de reconstruire dans l'espace image l'empreinte 2D E_{G_i} de la projection du noyau de reconstruction associé à chaque échantillon. L'image finale produite est donc basée sur un mélange des contributions des empreintes de chaque noyau de reconstruction. La figure 3.13 résume ce principe. L'avantage indéniable de cette méthode repose sur sa représentation sous forme d'un nuage de points. En effet, seuls les éléments de volumes utiles dans l'image finale sont modélisés, ce qui permet de réduire les données à traiter [109].

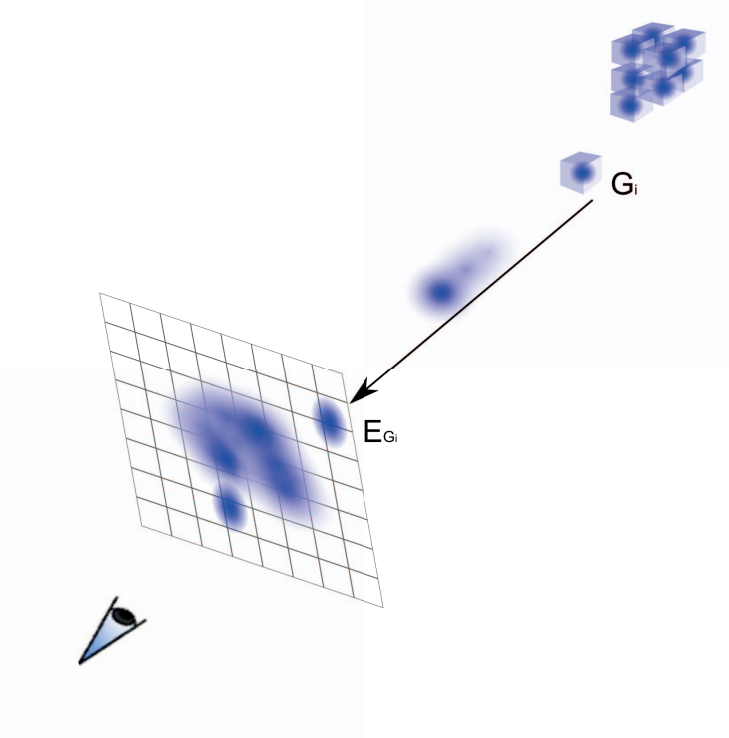


FIGURE 3.13 – Principe du *splatting* volumique : À chaque échantillon i du volume est attaché un noyau de reconstruction G_i (ici représenté par une fonction gaussienne). L'image finale est composée de la composition des empreintes E_{G_i} 2D résultant de la projection de chaque noyau sur le plan image.

Afin de résoudre le problème de visibilité et d'occlusion, il est généralement courant de trier les échantillons en fonction de leur distance à la caméra et de les composer dans l'ordre *front-to-back*. Ceci pose un problème de complexité pour les modèles détaillés et cause une parallélisation difficile de l'algorithme (ce qui conduit à diminuer l'efficacité d'une implémentation GPU). Une idée largement admise [164] est de regrouper les échantillons en « tranches » parallèles au plan image et de considérer tous les échantillons d'une même tranche comme étant situés à la même profondeur (voir figure 3.14). L'image finale est alors calculée comme la composition de toutes les tranches (chaque tranche peut être considérée comme une image intermédiaire). La qualité et la performance de cette approximation du tri repose naturellement sur le nombre de tranches utilisées.

De nombreux algorithmes [109, 110, 113, 114] de *splatting* ont été proposés pour améliorer la qualité de l'image générée, corrigeant la perspective ou ajoutant un filtre dans l'espace image de meilleure qualité. Cette méthode a de plus

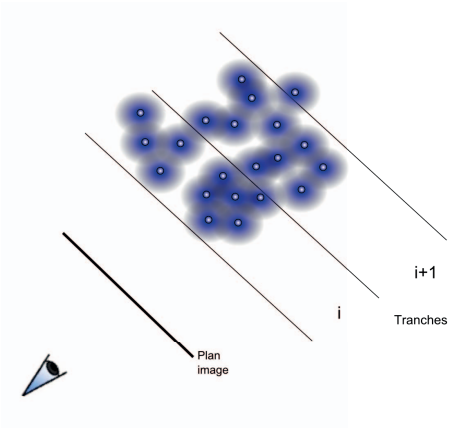


FIGURE 3.14 – Afin de combiner l'information de visibilité et une parallélisation efficace, l'idée admise est de regrouper les échantillons en tranches. Les contributions des échantillons de la tranche i sont mélangées.

été largement adaptée au matériel graphique. Xue et al. [171] comparent plusieurs implémentations GPU et discutent de l'accélération matérielle de différentes méthodes de *splatting*.

Un modèle populaire est le modèle d'*EWA volume splatting* (de l'anglais *Elliptical Weighted Average*) [131, 175]. Il propose un environnement permettant une visualisation interactive par méthode de splatting. Les auteurs proposent principalement de combiner à la projection du filtre de reconstruction de chaque noyau (dans ce cas une gaussienne tronquée) un filtre passe-bas dans l'espace image, afin de limiter les artefacts d'aliasing. Ce principe est présenté sur la figure 3.15.

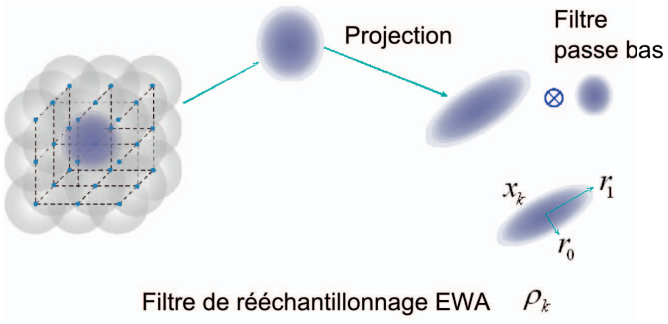


FIGURE 3.15 – EWA volume splatting. L'empreinte de chaque noyau de reconstruction est combinée avec un filtre passe-bas afin de limiter les artefacts d'aliasing. *Images de [25].*

Concrètement, l'implémentation matérielle de cette méthode consiste à projeter tous les échantillons appartenant à une tranche dans l'espace image et à générer par rasterisation un carré (dans l'espace image). Chaque fragment est ensuite testé afin de savoir s'il appartient effectivement à l'empreinte de la projection du noyau de reconstruction associé à son échantillon. La couleur déduite de chaque fragment est ensuite pondérée par un filtre de reconstruction et d'anti-aliasing et accumulée dans une image intermédiaire. Chen et al. proposent [25] de plus l'utilisation d'un filtre adaptatif, en fonction de la distance de l'échantillon à la caméra, afin de minimiser les coûteux calculs liés à l'évaluation du filtre. Toutefois, afin de résoudre le problème de visibilité, les auteurs proposent d'utiliser comme géométrie intermédiaire une grille régulière. Ceci impose une contrainte sur les données qu'il est possible de représenter grâce à cette méthode.

Il existe de nombreuses autres méthodes basées sur un principe de projection des éléments, comme par exemple les méthodes de projection de cellules (*cell projection*) [167], mais cela dépasse le cadre de cette thèse.

3.4 Algorithmes d'ordre image

Le principe de ces méthodes n'est pas de déterminer quelles sont les contributions que les objets de la scène apportent à l'image en réalisant leur projection. L'idée est de procéder de manière inverse, c'est-à-dire de considérer chaque pixel et de déterminer quelle est sa couleur finale en fonction des éléments correspondants dans la scène. L'idée est de suivre le cheminement inverse des rayons lumineux en lançant des rayons à travers l'image et de calculer la couleur de chaque pixel, remontant ainsi le chemin de la lumière parvenant à la caméra.

Le lancer de rayons [3, 165, 55] est la principale méthode d'ordre image. Le principe est de lancer un rayon pour chaque pixel à travers la scène afin de déterminer la visibilité des primitives. Ces rayons sont appelés *rayons primaires*. La primitive intersectant ce rayon la plus proche de l'utilisateur déterminera la couleur du pixel considéré. L'évaluation des interactions lumineuses avec la microstructure (voir section 2.3) permet de calculer la couleur finale du pixel. Cette méthode est appelée dans la littérature *ray casting*.

Cependant, cette méthode peut être améliorée pour prendre en compte les propriétés de réfractions et de réflexions des primitives. Dans ce cas, afin de calculer la couleur d'un pixel, un rayon réfléchi et/ou réfracté est lancé de-

puis la primitive intersectée par le rayon primaire (en respectant les propriétés optiques des matériaux (section 2.3)). La primitive ayant l'intersection la plus proche avec ces nouveaux rayons (*rayons secondaires*) ajoute sa contribution au pixel final, et génère également un ou plusieurs rayons secondaires. Cette approche récursive (*ray tracing*) permet de simuler toute l'optique géométrique et permet le rendu d'images photo-réalistes.
Cependant, le principe du lancer de rayons consiste à approximer l'intégrale de l'équation du rendu [64] en utilisant un seul rayon. Une meilleure approximation peut être obtenue en lançant un ensemble de rayons dans des directions aléatoires et en moyennant les contributions (en suivant le principe de l'intégration de Monte Carlo). Cette technique stochastique (*stochastic ray tracing*) est une approximation plus fidèle des phénomènes lumineux d'une scène et permet de gérer divers effets, tels les ombres douces, la profondeur de champ, l'éclairage indirect... Ces techniques produisent des résultats photo-réalistes mais requièrent énormément de temps de calcul et peuvent demander plusieurs heures pour générer une image en raison du nombre de rayons nécessaires.

La manière dont les interactions entre un rayon et une primitive sont évaluées dépend du type de la primitive. Nous présentons dans la section suivante (3.4.1) la gestion des primitives définies de manière surfacique, avant de présenter la gestion de primitives volumiques (section 3.4.2). Les dernières sections de ce chapitre concernent une rapide présentation de quelques améliorations et structures accélératrices des méthodes de lancer de rayon (section 3.4.3) ainsi que les détails d'une implémentation matérielle de ces méthodes sur les cartes graphiques (section 3.4.4).

3.4.1 Gestion de surfaces

Pour traiter les éléments définis de manière surfacique par leur bord, nous reprendrons l'hypothèse généralement admise de l'influence nulle du vide sur le rayon de vue. Dans cette optique, il n'est pas nécessaire de calculer l'interaction de la scène en chaque point du rayon de vue. Il s'agit donc simplement de calculer l'intersection entre un rayon de vue et chaque objet de la scène.

Le calcul d'intersection entre un rayon $\vec{V}$ et une primitive de la scène dépend du type de la primitive :

Primitive paramétrique Le calcul de l'intersection entre un rayon de vue et une primitive paramétrique est un problème difficile et non trivial. Ces travaux dépassant le cadre de cette thèse, nous renvoyons le lecteur vers le tour d'horizon proposé par Han et al. [59] .

Primitive implicite Ce problème d'intersection se résume souvent à trouver les zéros d'une équation polynomiale (c'est-à-dire le point où la distance entre le rayon V et la primitive paramétrique est nulle). Pour les primitives les plus simples, cette équation est parfois résoluble directement. Les primitives plus complexes, comme le tore par exemple, requièrent des méthodes itératives plus complexes pour approximer la solution, comme la méthode de Newton-Raphson [53], les séquence de Sturm [153], comme proposé par De Toledo [149]. De nombreux travaux [70, 150] proposent également d'utiliser l'arithmétique affine des intervalles pour effectuer le rendu de primitives implicites par lancer de rayon.

Primitive explicite Dans le cas de primitives explicites surfaciques, comme par exemple une primitive polygonale, la simplification courante consiste à se ramener au problème simple de l'intersection entre un rayon et un triangle. En effet, tout polygone peut être décomposé en un ensemble de triangles. L'intersection peut se résumer à calculer l'intersection entre le rayon de vue et le plan du triangle et à déterminer si le point résultant (sur le plan du triangle) se trouve à l'intérieur ou à l'extérieur du triangle [10].

Une fois que ce point d'intersection est déterminé, il suffit de calculer la contribution de ce point au pixel final, en lançant éventuellement d'autres rayons dans la scène et en traitant à nouveau la première intersection rencontrée le long de ce rayon. En règle générale, afin d'obtenir un ombrage correct (voir section 2.3), il est également nécessaire de définir la normale à la surface au point d'intersection.

3.4.2 Gestion de volumes

Comme on l'a vu dans la section 3.1.2, on ne peut se contenter d'une simple recherche d'intersection lorsqu'on veut afficher un modèle totalement volumique. Le principe d'une méthode de lancer de rayon pour une scène volumique est d'évaluer l'intégrale du rayon volumique (équation 3.3) le long du rayon de vue. L'idée est donc d'échantillonner le rayon de vue, d'évaluer la contribution de la scène à chaque échantillon et de composer ces contributions. Cette technique dérivée du lancer de rayon est appelée *ray marching* et est illustrée sur la figure 3.16. Les échantillons successifs sont combinés entre eux en utilisant l'approximation courante (équation 3.9) décrite dans la section 3.1.2.

Cette technique est valable pour un rendu volumique direct indépendamment de la représentation des données volumiques. En effet, cette méthode

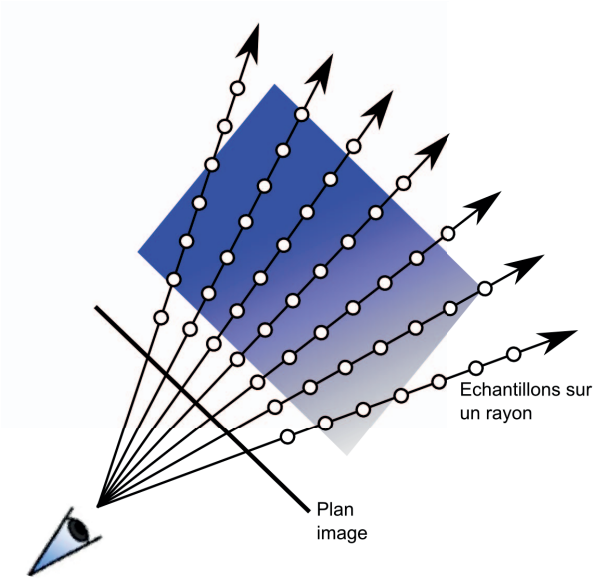


FIGURE 3.16 – Principe du ray marching. Un rayon échantillonné est généré pour chaque pixel. L'intégrale du rendu volumique est évaluée à chaque échantillon afin de reconstruire l'image finale.

suppose simplement qu'il est possible d'évaluer les propriétés optiques d'un échantillon situé le long d'un rayon de vue. La qualité et les performances d'une méthode de rendu volumique par *ray marching* sont bien entendu dépendantes de la précision de cette évaluation et donc de la qualité du filtre de reconstruction utilisé (dans le cas de données définies de manière explicite par exemple) ainsi que du pas d'échantillonnage.
De manière générale, le taux d'échantillonnage du rayon de vue a une importance cruciale sur la méthode de *ray marching*. Intuitivement, l'évaluation de l'intégrale du rendu volumique reposant sur une discrétisation, un taux d'échantillonnage élevé produira une approximation plus fidèle et donc des images de meilleure qualité, au détriment toutefois des performances. Il est à noter qu'un échantillonnage régulier n'est pas obligatoire dans ce cas. En effet, de nombreuses méthodes [89] proposent un échantillonnage adaptatif, de manière à accroître les performances sans diminuer la qualité des images générées.

3.4.3 Structures accélératrices et autres simplifications

Il existe plusieurs techniques pour optimiser les méthodes de lancer de rayons en éliminant les calculs inutiles ou redondants. Il est courant par exemple de partitionner la scène afin d'effectuer les calculs d'intersection uniquement

avec les objets présents dans la partie de la scène traversée par le rayon. Une idée similaire consiste à entourer chaque objet complexe d'une boîte englobante (offrant un calcul d'intersection simple) et d'effectuer le calcul d'intersection (coûteux) d'un rayon avec l'objet complexe uniquement si le rayon intersecte la boîte englobante.
Un schéma d'accélération populaire est également d'arrêter au plus tôt les calculs le long d'un rayon ayant atteint une opacité maximale (*early ray termination* en anglais). En effet, les contributions successives sur un tel rayon seront masquées par les contributions précédentes (voir équation 3.9).
On peut également citer l'exploitation de la cohérence spatiale [155] [132] [135] comme schéma d'accélération des méthodes de lancer de rayon. L'idée est de grouper les rayons émanant de pixels voisins par paquets ou cônes, afin de factoriser les calculs.

En ce qui concerne la qualité des images générées par lancer de rayons, une approche courante consiste à diminuer les artefacts d'aliassage en augmentant le nombre de rayons lancés par pixel et en moyennant les contributions.

3.4.4 Lancer de rayon sur GPU

Les algorithmes de lancer de rayon ont vu leurs performances devenir interactives pour des scènes simples grâce à leur implémentation sur le matériel graphique. En effet, étant par nature parallélisables, les algorithmes de lancer de rayon bénéficient directement des capacités de calcul croissantes des cartes graphiques. Les méthodes effectuant un lancer de rayon assisté par le GPU ont en commun l'utilisation des capacités de rastérisation de la carte graphique pour générer un rayon par pixel en utilisant une géométrie de substitution, et l'implémentation au sein du *fragment program* des recherches d'intersection rayon-primitives. Toutefois, les structures accélératrices classiques en lancer de rayon requièrent plus d'effort pour être implémentées totalement sur le GPU, en raison des spécificités de l'architecture du matériel graphique. Dans cette optique, Carr et al. [22] présentent une méthode mixte pour le rendu de maillages triangulaires. Cette technique se repose le GPU pour calculer les intersections rayons-triangles et utilise le CPU pour traverser efficacement les structures accélératrices. Leur méthode génère toutefois d'importants transferts de données entre le CPU et le GPU, ce qui limite les performances. Une autre méthode entièrement GPU pour représenter des maillages triangulaires est proposée par Purcell et al [128]. Dans leur approche, les auteurs utilisent plusieurs passes de rendu pour générer les rayons, calculer les intersections et effectuer l'ombrage des éléments directement sur le GPU.

Les méthodes de rendu de surfaces implicites par lancer de rayon bénéficient également d'une implémentation matérielle. De Toledo [148] propose l'implémentation sur le GPU de divers algorithmes de recherche d'intersection entre un rayon et une primitive implicite. Il démontre également que, lors du rendu d'un grand nombre de primitives implicites (des tores par exemple), un rendu basé sur une implémentation GPU d'un lancer de rayon est plus rapide qu'une méthode de projection/rasterisation des mêmes surfaces représentés par des polygones. Knoll et al. [70] proposent également une méthode robuste reposant sur l'utilisation de l'arithmétique affine des intervalles pour le calcul d'intersection avec une primitive implicite de degré arbitraire.

Les données définies de manière volumique tirent également parti de telles implémentations GPU. Par exemple, Roettger et al. [133] démontrent la pertinence d'un lancer de rayon GPU par rapport au méthodes à base de *slicing*. Ils proposent notamment l'utilisation d'un schéma d'échantillonnage adaptatif le long d'un rayon de vue. Dans la même idée, Kraus et al. proposent [74] un schéma d'échantillonnage adaptatif dans plusieurs directions (et non plus seulement le long du rayon de vue) afin de réduire encore le nombre d'échantillons nécessaires sans dégrader la qualité des images générées.

L'implémentation d'une méthode de rendu volumique par lancer de rayon sur le GPU repose principalement sur l'utilisation conjointe des propriétés de rasterisation et des capacités de traitement par pixel du matériel graphique [76] [142]. En règle générale, les données volumiques sont stockées directement sur la carte graphique sous la forme d'une texture 3D. Afin de réaliser le rendu de ces données, un cube englobant les données est projeté sur l'écran (par le *vertex program*). L'idée est d'utiliser cette étape pour calculer les coordonnées et la direction des rayons de vue traversant le volume. À chaque coin du cube sont stockées les coordonnées 3D correspondant à la position du sommet dans le repère du cube. Une première projection des faces arrières du cube permet de générer un ensemble de fragments ayant en paramètre les coordonnées 3D correspondant au point de sortie du rayon de vue du cube (unité). Une phase identique projetant les faces avant du cube permet de récupérer les coordonnées d'entrée du rayon de vue dans le cube (voir figure 3.17). Ces coordonnées (exprimées dans le repère du cube unité) peuvent alors directement être utilisées pour indexer la texture 3D et ainsi fournir la définition d'un rayon de vue traversant les données volumiques. Il ne reste plus qu'a échantillonner le rayon afin de procéder à l'évaluation de l'intégrale du rendu volumique. Cette technique, reprise dans nos travaux, est détaillée plus avant à la section 5.1.1.

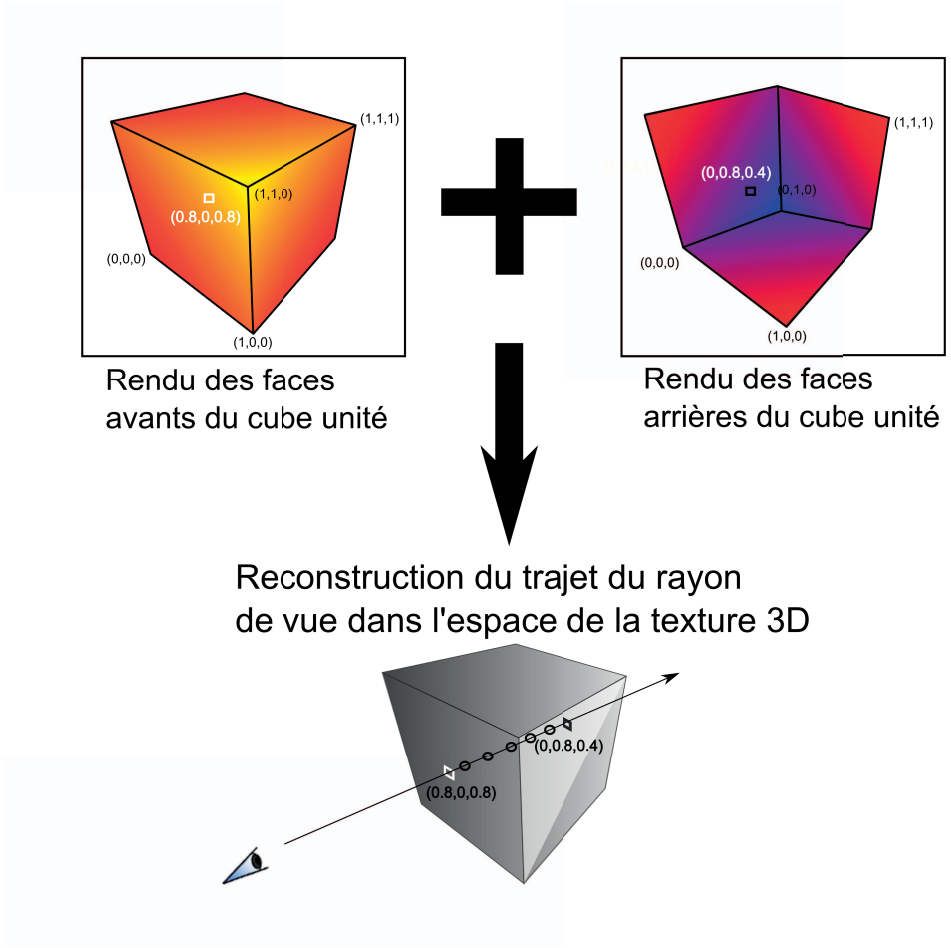


FIGURE 3.17 – La rastérisation des faces arrières et avants du cube unité, aux sommets duquel sont rattachées des coordonnées 3D, permet de reconstruire dans l'espace de la texture 3D le trajet du rayon de vue.

Conclusion

Nous avons vu dans ce chapitre les grands principes des majeures méthodes d'affichage, ainsi que leurs implémentations sur le matériel graphique. Ces techniques de rendu présentent chacune des avantages et des défauts particuliers et sont plus ou moins adaptées au rendu des différentes primitives. Une des pierres angulaires de la majorité des moteurs 3D dans l'industrie repose sur les capacités de rastérisation du GPU pour les maillages triangulaires. Des détails visuels sont ajoutés par l'utilisation conjointe de textures 2D et de l'évaluation des contributions lumineuses par le modèle de Phong. Toutefois, ces techniques montrent leurs limites dans le cas de scènes complexes. En effet, dans le cas d'objets présentant une apparence très détaillée, la complexité générée par une modélisation explicite est trop importante pour permettre un rendu interactif. Afin de réduire cette complexité, un niveau de représentation intermédiaire est introduit, appelé « mésostructure ». Ce niveau de représentation concerne les détails d'un modèle n'ayant pas d'influence globale sur sa forme. Nous nous concentrons dans ces travaux sur le rendu de mésostructures et présentons dans le chapitre suivant des méthodes hybrides se focalisant sur l'affichage efficace de mésostructures.

Chapitre 4

Rendu de mésostructures : méthodes existantes

Comme mentionné dans l'introduction, la décomposition d'un modèle en différents niveaux structurels est une technique bien connue dans le domaine de l'informatique graphique et permet d'augmenter le réalisme des images de synthèse générées. Nous présentons dans ce chapitre les méthodes existantes abordant le problème du rendu de mésostructures. Lorsque l'on s'intéresse au rendu de mésostructures, il convient de se poser la question suivante : quels sont les phénomènes responsables du réalisme final des images résultantes ? Nous pouvons dégager plusieurs phénomènes, souvent difficiles à reproduire, mais contribuant de manière significative à l'aspect réaliste des images représentant des mésostructures :

La silhouette. Lorsqu'un modèle est vu avec un angle rasant, il est important de voir la silhouette de la mésostructure se découper sur le fond sous peine de réduire le réalisme de la scène.

La parallaxe. La parallaxe est un phénomène que l'on peut aussi exprimer sous le terme d'auto-occlusion, c'est-à-dire le fait de masquer un élément de la mésostructure par un autre élément de la même mésostructure. Alors que le phénomène d'occlusion est traité de manière efficace par le matériel graphique pour une échelle macrostructurelle, il est parfois difficile de reproduire cet effet dans le cadre d'un rendu de mésostructure.

Auto-ombrage et inter-réflexions. Le problème d'ombrage est inhérent à la simplification courante consistant à considérer l'illumination d'un point de vue local (voir section 2.3.3). À l'instar de la parallaxe, des méthodes permettent d'approximer le phénomène d'ombrage au niveau macrostructurel. Il reste toutefois plus difficile de reproduire ces effets à un niveau mésostructurel.

Cette liste, bien entendu non exhaustive, regroupe quelques phénomènes que nous considérons importants pour le réalisme des mésostructures des ima-

ges de synthèse. Il est important de considérer chacun de ces points afin de discuter de la pertinence d'une méthode de rendu de mésostructure. En se plaçant du point de vue des techniques de rendu, nous pouvons rajouter trois critères importants :

Performances. Ce problème, posé par le temps de calcul et donc la vitesse de l'affichage de la mésostructure, ne joue pas dans le réalisme des images de synthèse générées mais reste néanmoins un point important à considérer lors de la création d'une méthode de rendu.

Coût mémoire. Alors qu'aujourd'hui le matériel graphique est indispensable pour obtenir le rendu temps réel d'une scène complexe, le problème de l'occupation mémoire des données nécessaires à un algorithme de rendu se pose à nouveau. En effet, alors que les performances du matériel graphique augmentent de manière drastique, l'espace mémoire disponible sur la carte graphique reste plus limité.

Gamme des mésostructures. La question qui se pose ici est principalement de savoir quels sont les types de mésostructures qu'il est possible de représenter avec une technique de rendu. La figure 4.1 illustre avec des photographies quelques gammes de mésostructures naturelles complexes. La question (ouverte) sous-jacente à ce point est : « est-il nécessaire d'avoir une méthode générale pour tous les types de mésostructures ou est-il plus pertinent d'avoir une technique de rendu spécialisée (et généralement plus efficace) pour différentes classes de mésostructures ? »

Après ces quelques points qu'il est nécessaire de garder à l'esprit en considérant les techniques de rendu réalistes de mésostructure, nous présentons dans ce chapitre un bref état de l'art sur les différentes techniques existantes.

4.1 Techniques de modélisation et de visualisation de mésostructures

Nous rappelons dans cette section les différentes méthodes existantes permettant le rendu de mésostructure pour un objet quelconque. Nous allons pour cela distinguer deux approches : l'approche surfacique (section 4.1.1) et l'approche volumique (section 4.1.2). La première approche consiste à considérer uniquement un espace 2D situé à la surface du modèle (espace de paramétrisation de la surface). Le rendu de mésostructure en lui même est effectué dans cet espace de manière indépendante ou quasi-indépendante de l'objet dans la majeure partie des cas. L'approche volumique quant à elle, vise à ajouter des détails de mésostructure dans l'espace 3D du modèle.

FIGURE 4.1 – Exemples naturels de mésostructures complexes.

4.1.1 Approche surfacique

L'idée d'une approche surfacique consiste à paramétrer un modèle défini de manière surfacique et à se placer dans l'espace 2D de cette paramétrisation afin de calculer l'apparence de la mésostructure. Nous allons présenter dans cette section un ensemble de méthodes utilisant cette approche.

Placage de texture

La représentation la plus simple pour la visualisation de détails à la surface d'un objet consiste en l'application d'une carte de couleur, appelée texture 2D, comme présenté dans la section 2.4. Cette représentation bidimensionnelle se plaque à la surface d'un modèle et permet de définir une couleur arbitraire pour chaque fragment composant la surface du modèle. Cette méthode pose

le problème de la définition d'une correspondance entre un point 3D de la surface du modèle et un texel (élément de texture, voir section 2.4). Ce problème de placage de textures induit en outre une distorsion de l'apparence, ce qui dans certains cas engendre des artefacts visuels.
Il est rapidement apparu que l'utilisation d'une texture solide (c'est-à-dire définie dans un espace 3D au lieu de 2D) permet de se libérer de ce problème, comme proposé par Peachey et al. [118] et Gardner [50][51]. En effet, ces textures évitent la définition d'une paramétrisation. Il suffit de faire correspondre les coordonnées d'un point 3D à la surface d'un modèle à un point 3D dans l'espace de définition de la texture. Les textures solides ne présentent pas d'artefacts de distorsion. L'objet semble alors « taillé » directement dans un bloc solide, comme illustré sur la figure 4.2.

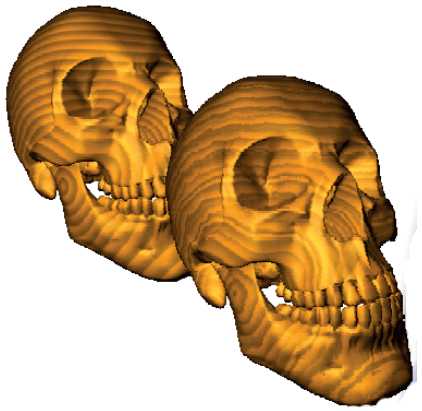
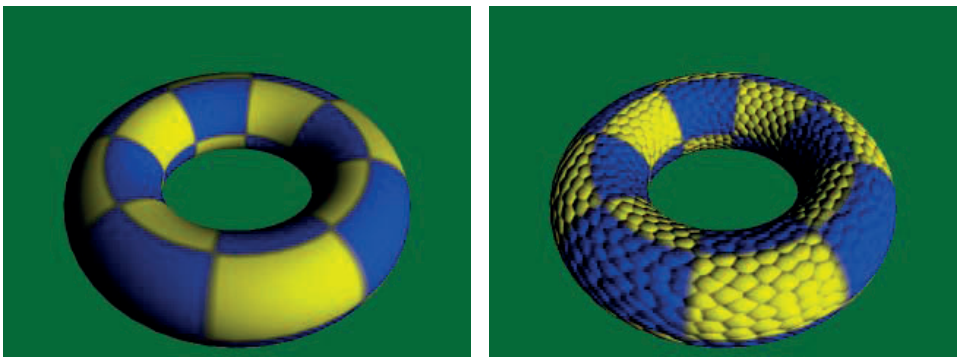


FIGURE 4.2 – Objets avec une texture 3D solide. L'objet semble taillé dans un matériau.

Ces techniques d'application de textures de couleur à la surface d'un objet sont largement utilisées dans le domaine de la synthèse d'image, d'une part pour leur simplicité et leur intuitivité, et d'autre part pour leurs performances, le matériel graphique étant largement optimisé pour un tel type de technique. Toutefois, la gamme de mésostructures pouvant être représentée de manière réaliste par ces techniques reste très fortement limitée : les effets liés à la géométrie nécessaires au réalisme ne peuvent pas être représentés par ce type d'approche. En effet, ces techniques ajoutent simplement une information de couleur à la surface des objets et ne permettent pas de prendre en compte les divers phénomènes que nous avons présentés au début de ce chapitre, tels les effets de silhouette, de parallaxe, ou d'éclairage.

Placage de mésostructure par « perturbation » de normale

L'application d'une texture de couleur à un objet permet de représenter une variation des propriétés colorimétriques d'une surface de manière efficace. Mais l'utilisation d'une simple technique de placage de texture de couleur ne permet pas de représenter des reliefs ou des structures géométriques complexes.



(a) Carte de couleur (b) Bump Mapping

FIGURE 4.3 – L'aspect lisse d'une mésostructure rendue avec une simple texture de couleur (à gauche) peut être partiellement corrigé par l'utilisation du *bump mapping* (à droite).

Afin d'étendre le champ d'application du placage de texture, Blinn [16] propose la technique du *bump mapping*. Cette technique donne une impression de relief, c'est-à-dire de creux et de bosses à la surface d'un objet. Cette approche ne rajoute pas de détails à l'objet en lui-même mais donne l'illusion d'une mésostructure complexe en modifiant la normale utilisée lors de l'évaluation de l'équation d'illumination (équation 3.2). La normale à la surface peut être perturbée par une fonction aléatoire, ou remplacée par une normale issue d'une *texture de normales* (*normal mapping*) [27] [29] . Cette texture ne donne pas une information de couleur comme les textures traditionnelles, mais définit en chaque point de la surface une nouvelle normale remplaçant la normale de la surface de l'objet [1]. Cette normale perturbée est utilisée au cours de l'évaluation de l'équation d'illumination, créant ainsi un effet visuel de relief à la surface de l'objet. Un exemple de *bump mapping* est présenté dans la figure 4.3. Cette méthode permet d'augmenter la gamme de mésostructure qu'il est possible de représenter en utilisant une technique de placage. Cependant, de nombreux effets, tels les problèmes de silhouettes ou de parallaxe ne sont

1. Une normale étant un vecteur 3D, elle peut donc être stockée dans une texture RVB sur le matériel graphique.

pas directement pris en compte avec cette technique. Le problème de l'auto-ombrage pour le placage de mésostructure a été traité par la méthode d'*horizon mapping* [104] [140], qui se base sur un précalcul. De façon similaire, Ashikhmin et al. [6] proposent le précalcul et la reconstruction au vol des interactions lumineuses pour une surface possédant un faible relief. De la même manière, Heidrich et al [61] utilisent des textures pour stocker les informations de visibilité et d'interactions lumineuses de la mésostructure avant de reconstruire son apparence finale, prenant ainsi en compte les effets de l'illumination indirecte et des ombres. L'ensemble de ces méthodes ne permet toutefois pas de représenter les silhouettes de la mésostructure et ne permet de représenter qu'une gamme restreinte de mésostructures (définissables par une carte de hauteurs).

Méthodes basées images

Une idée classique dans le domaine de l'informatique graphique consiste à utiliser un ensemble d'images (représentant la mésostructure vue sous des conditions particulières) et d'utiliser cet ensemble lors du rendu pour restituer l'apparence de la mésostructure. Un tel type d'approche consiste dans notre cas à échantillonner l'apparence de la surface sous diverses directions de vue et d'illumintation. Les travaux de Dana et al. [34] sur les *Bidirectionnal Texture Functions* font partie de cette catégorie de représentation. L'idée est de capturer l'apparence de la mésostructure (à l'aide de matériel de mesure) selon des conditions d'éclairage et de vue précises et de stocker ces représentations directement en mémoire graphique. Lors du rendu de la surface de l'objet, la représentation capturée avec les conditions de vue et d'illumination correspondant aux paramètres de la scène sera choisie et utilisée. En pratique, l'image utilisée pour l'affichage sera créée par interpolation entre les représentations ayant les paramètres d'acquisition les plus proches de ceux de la scène synthétique. Cette structure permet d'associer à un point de la surface de l'objet, soumis à certaines conditions de vue et d'éclairage, une couleur finale prenant en compte divers effets tels les inter-réflexions, l'auto-ombrage et les phénomènes de parallaxe. En pratique, cette méthode est une fonction 6D :

$$c_{BTF}(u, v, \theta_{in}, \phi_{in}, \theta_{out}, \phi_{out}) \tag{4.1}$$

avec (u, v) les coordonnées de texture du pixel considéré, (θ_{in}, ϕ_{in}) la direction de la source lumineuse et $(\theta_{out}, \phi_{out})$ la direction de vue.

Cette technique apporte des résultats visuellement convaincants, comme illustré dans la figure 4.4, mais elle n'est pas dénuée de certains problèmes. De prime abord, le problème majeur soulevé par ces méthodes est celui de l'espace

FIGURE 4.4 – Les BTF [34] permettent la reproduction fidèle de nombreux effets lumineux. Ici, une comparaison entre un placage de texture (à gauche) et l'application d'une BTF (à droite) *Images de [108]*

mémoire nécessaire à une telle représentation. En effet, la représentation efficace des phénomènes de mésostructure cités en début de chapitre nécessite un échantillonnage relativement fin des directions d'illumination et de vue (spécialement pour des mésostructures complexes ou avec des parties saillantes). Ceci augmente de manière significative la taille mémoire nécessaire pour le stockage de cette représentation.
De nombreux travaux [108] ont été menés, notamment dans le but de réduire le coût mémoire de cette approche, par exemple en utilisant une compression de données basée sur une analyse en composantes principales [72][154]. Müller et al.[111] proposent également une partition par pixel et obtiennent un taux de compression de 100 :1 tout en gardant des performances temps réel lors du rendu. Toutefois, la plupart de ces méthodes peuvent difficilement afficher plus de quelques petits échantillons de mésostructure répétés sur un objet. Il est également difficile avec cette méthode de prendre en compte l'influence de la courbure de l'objet sur les effets de parallaxe, ou de représenter la silhouette de la mesostructure.

De manière plus générale, cette méthode souffre d'un problème commun aux méthodes basées sur des ensembles d'images. Lorsque l'échantillonnage des directions de vue est trop restreint pour capturer fidèlement les détails de la mésostructure, deux représentations capturées avec des directions de vue voisines peuvent avoir une apparence très différente. Une interpolation entre ces deux représentations produira un résultat exhibant un effet d'apparition et de disparition des éléments (l'effet de « *ghosting* »).

Afin de réduire la complexité de la représentation à base de BTF (de dimension 6), différentes méthodes visent à réduire leur dimensionalité en fixant certains paramètres. Nous pouvons dégager deux types majeurs de simplification des BTF : les méthodes échantillonnant uniquement l'éclairage et les méthodes capturant la mésostructure sous différentes directions de vue.

– Kautz et al. [69] proposent de simplifier la représentation des BTF en fixant le point de vue et en capturant la mésostructure sous différentes conditions d'illumination. Le principe est de réduire le nombre d'images nécéssaires (par rapport à une représentation basée sur les BTF) pour représenter la mésostructure. Pour ce faire, les auteurs proposent de considérer la radiance moyenne sortante sur l'ensemble de la mésostructure et de ne garder qu'une unique représentation pour chaque valeur de radiance sortante. Cela implique dans un premier temps de négliger les effets de masquage et d'auto-ombrage. Lors d'un rendu, l'utilisation d'une BRDF arbitraire pour l'échantillon de surface représentant la mésostructure permet de calculer une valeur de radiance sortante « globale » qui sera utilisée pour choisir la représentation utilisée. Cette méthode permet la réduction effective du nombre d'images nécessaire pour représenter la mésostructure et donne des résultats convaincants. Toutefois, cette méthode souffre de son caractère totalement empirique. En effet, le choix de la représentation finale repose totalement sur la valeur de radiance calculée par la BRDF choisie de manière arbitraire lors du rendu. De ce fait, la représentation fidèle de la mésostructure est dépendante du choix empirique de la BRDF (qui doit refléter le comportement moyen de la mésostructure lors des interactions lumineuses).

  Les travaux de Malzbender et al. [101] s'inscrivent également dans l'optique de réduction de dimensionalité des BTF et proposent une simplification similaire par l'introdution des *Polynomial Texture Maps* (PTM). Ces travaux intègrent également une technique de stockage consistant à séparer les informations de luminance et de chrominance de la mésostructure, en introduisant une représentation biquadratique liant la luminance à la direction de la lumière incidente. Cette méthode permet la reproduction fidèle de nombreux effets lumineux (Fresnel par exemple) et produit des images réalistes. Cependant, la méthode reste contrainte par son unique point de vue. De plus, le protocole d'acquisition reste contraignant et limite de ce fait l'utilisation de cette méthode. A l'instar des BTF, cette méthode ne prend pas en compte la silhouette de la mésostructure. De par son hypothèse (une direction de vue fixe), cette mé-

thode ignore également les effets de parallaxe. De plus, on observe avec cette technique un fort lissage des effets lumineux spéculaires dus à la représentation bi-quadratique.

– La simplification complémentaire à ce problème de dimensionalité des BTF est d'acquérir l'apparence de la mésostructure sous différents points de vue. C'est ce que proposent Wang et al. [156] au travers des *view-dependant displacement maps* (VDM). Cette approche a pour idée principale de sto-

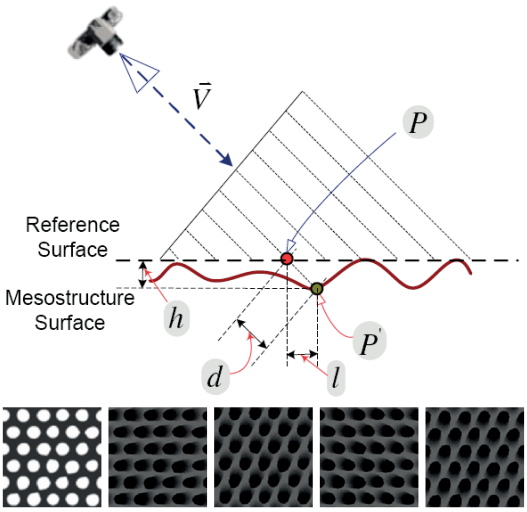


FIGURE 4.5 – *View-dependent Displacement Mapping* [156]. En haut : Le principe de la méthode. Pour chaque point de vue $\vec{V}$, on stocke, pour chaque point P de la surface de réference, la distance l à la mésostructure. En bas : Un ensemble de texture contenant pour chaque texel la distance à la mésostructure pour la direction de vue $\vec{V}$. *Images de [156]*

cker dans un ensemble de textures 2D non pas une information de couleur mais une information de « profondeur » le long d'un rayon de vue. L'idée est ici de capturer uniquement les effets de parallaxe et de silhouette au prix des interactions lumineuses. Comme présenté sur la figure 4.5, il s'agit pour chaque point P de la surface d'un objet, de calculer la distance au point le plus proche P' de la mésostructure dans la direction de vue $\vec{V}$ et de stocker le résultat dans une texture 2D correspondante. Grâce à ce jeu de textures, l'emplacement du point P' de la mésostructure peut être calculé rapidement lors du rendu (en sélectionnant la texture 2D correspondant à la direction de vue). Toutefois, cette méthode de rendu reste valide uniquement pour les objets plans. En effet, comme illustré par la figure 4.6, lorsque l'on applique cette méthode à des objets présentant une courbure, les points P' calculés ne correspondent plus à l'intersec-

tion entre le rayon de vue et la mésostructure.

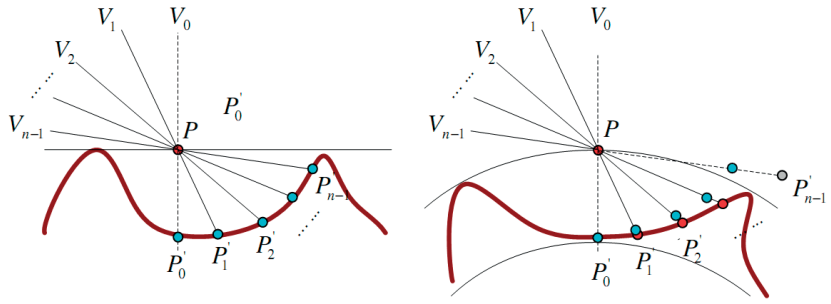


FIGURE 4.6 – *Les surfaces incurvées causent des erreurs lors de la reconstruction de la mésostructure. Images de [156]*

Afin de prendre en compte ces effets de distortions, il convient de considérer des informations supplémentaires. En effet, plutôt que de calculer pour chaque fragment représentant la surface d'un objet une direction de vue particulière, ce qui poserait des problèmes d'intégration sur certains matériels graphiques, les auteurs proposent d'utiliser une information de courbure c. Ce paramètre permet de corriger lors du rendu l'influence de la forme de l'objet en lui-même sur la mésostructure. Ceci revient à créer une structure 5D permettant d'associer une valeur de déplacement à chaque pixel de l'objet :

$$d_{\mathrm{VDM}}(u, v, \theta, \phi, c) \tag{4.2}$$

avec (u, v) les coordonnées de textures, (θ, ϕ) les angles sphériques des directions de vue et c la courbure locale de l'objet.
L'utilisation de la courbure c et d'une valeur seuil autorisant l'élimination des fragments dont les rayons de vue n'intersectent pas la mésostructure, ceci permet d'approximer les effets de silhouettes, comme montré sur la figure 4.7.
Les effets d'auto-ombrage sont recalculés lors du rendu de manière efficace en remplaçant la direction de vue par la direction de la lumière. Une comparaison avec le champ de hauteurs correspondant permet de déterminer si la ligne entre la lumière et le pixel considéré est interrompue (ce pixel est alors dans l'ombre d'une autre partie de la mésostructure). Cette méthode donne des résultats intéressants mais souffre encore de quelques limitations. Premièrement, l'utilisation de cette technique implique un coût mémoire non négligeable dû au stockage exhaustif des

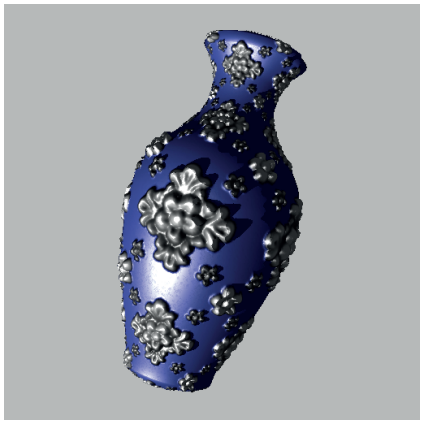


FIGURE 4.7 – La méthode de *View-dependent Displacement Mapping* permet de traiter de manière efficace les effets d'auto-ombrage et de silhouette. Images de [156]

informations de mésostructure. Ce coût mémoire peut rapidement se révéler problématique pour des mésostructure complexes (requiérant donc un échantillonage dense des directions de vue afin de ne pas manquer de détails), et ce malgré les méthodes de compression pouvant être appliquées [156]. Comme toute méthode basée images, il est à noter que la discrétisation des directions de vue peut générer des artefacts pour des mésostructures complexes ou très saillantes.

Modélisation explicite avec reconstruction au vol

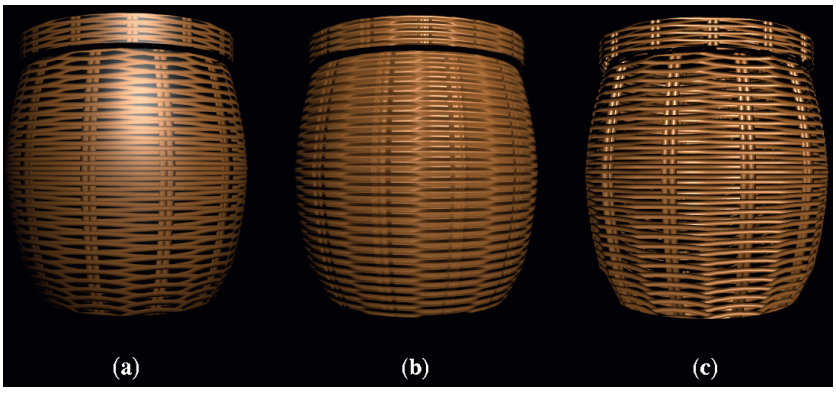


FIGURE 4.8 – Avantage du rendu de mésostructure par lancer de rayon. Sur ces images, la mésostructure est rendue par : (a) Texture 2D traditionnelle. (b) *bump mapping* (c) Lancer de rayon virtuel. Images de [41]

L'une des idées principales pour s'affranchir des problèmes liés au précalcul et à l'échantillonnage des directions de vue est naturellement de procéder à une reconstruction totale de l'apparence de l'objet lors du rendu. Le principe consiste alors, pour chaque fragment à la surface de l'objet, à « plonger » le rayon de vue issu de ce fragment dans le référentiel de la mésostructure de manière à calculer efficacement son apparence. Dischler [41] propose par exemple d'exprimer un rayon de vue dans le référentiel de la mésostructure (coïncidant ici avec le référentiel à la surface de l'objet au point considéré) et de calculer la première intersection avec la mésostructure. Cette approche permet de traiter les effets de parallaxe et de silhouette directement lors du rendu, comme illustré sur l'image 4.8. Cette méthode est réalisée entièrement sur le CPU, et utilise une structure accélératrice dépendante du point de vue. De plus, cette méthode, valide pour une surface plane, ne prend pas en compte les distortions induites par un objet présentant des courbures.

Avec les avancées du matériel graphique, l'idée est rapidement apparue d'effectuer un calcul similaire d'intersection mais avec une structure plus simple : un champ de hauteurs. En effet, cette représentation est simple et naturellement adaptée au matériel graphique (voir section 2.1.3). Il existe principalement deux catégories de techniques : les méthodes itératives et les méthodes non-itératives. Alors que les méthodes itératives se focalisent sur la recherche (itérative) de l'intersection exacte d'un rayon de vue avec la carte de hauteur, les méthodes non-itératives utilisent un calcul simple et rapide pour approximer grossièrement cette intersection.

– La technique du *parallax mapping*, introduite par Kaneko et al. [68], fait partie de cette dernière catégorie. Le principe est d'utiliser une valeur d'élévation définie par une carte de hauteurs pour approximer les effets de parallaxe de la mésostructure, en supposant une hauteur constante. La figure 4.9 illustre cette technique. A partir des coordonnées (u, v) d'un point de la surface visualisée, on accède à la hauteur h de l'élément de mésostructure correspondant. L'idée est de considérer que la mésostructure se comporte comme une surface plane parallèle à la macrostructure et de hauteur h. Il est alors possible de calculer l'intersection du rayon de vue V avec cette surface, donnant ainsi des nouvelles coordonnées u', v' qui permettront de déduire une couleur pour le point considéré. Cette méthode permet une approximation très rapide de la contribution de la mésostructure. Toutefois, cette approximation peut (dans le cas de mésostructures avec de grandes variations de hauteur) être très éloignée de la mésostructure en elle-même et donner lieu à des artefacts visuels.

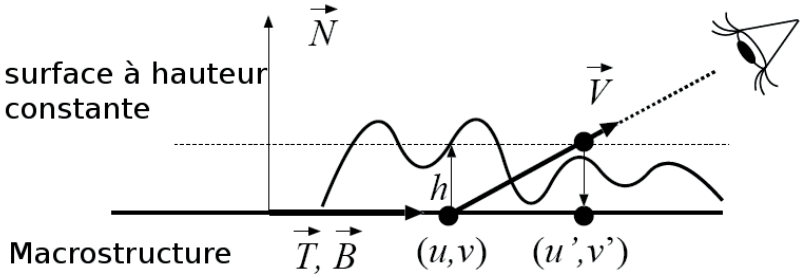


FIGURE 4.9 – Le parallax mapping. Le principe est d'utiliser une technique simple pour approximer grossièrement l'intersection entre le rayon de vue et la mésostructure. *Images de [146]*

Cette méthode offre effectivement des informations de parallaxe ainsi que d'excellentes performances, mais gère difficilement les angles rasants et les effets de masquage. De nombreuses extensions [147] [105] ont été proposées pour réduire ces artefacts, mais la méthode reste toutefois basée sur une approximation simple et ne peut traiter des mésostructures trop complexes.

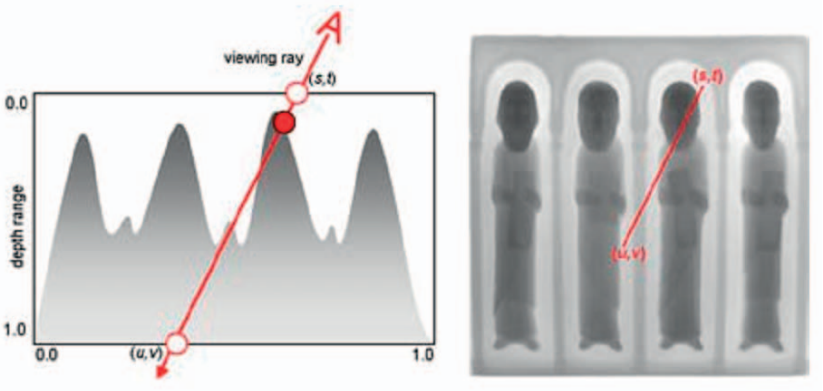


FIGURE 4.10 – La recherche d'intersection avec un champ de hauteurs (à gauche). La recherche est effectuée en 2D, à partir des coordonnées (s, t) et continue jusqu'a ce qu'une valeur inférieure à la valeur de hauteur le long du rayon de vue soit obtenue ou jusqu'a atteidnre (u, v) (à droite).*Images de [126]*

– Avec le développement des capacités de calcul du matériel graphique, les méthodes de rendu basées sur une recherche itérative de l'intersection d'un rayon de vue et de la mésostructure connaissent un essor non négligeable. Le *relief mapping* [116, 117, 47] est une méthode qui se classe

dans cette catégorie. La figure 4.10 illustre le principe de ces méthodes. Concrètement, cela revient à exprimer le rayon de vue (pour un fragment donné) comme un segment orienté dans l'espace 2D d'une texture représentant la mésostructure. En chaque point de ce segment, il est possible de définir une valeur d'élévation (une hauteur) par rapport à une surface de référence. Chaque texel de la texture contient également une valeur d'élévation (la hauteur de la mésostructure en ce point). Le rendu de la mésostructure consiste à déterminer le premier point le long du segment où la hauteur du segment est inférieure à la valeur d'élévation contenue en ce texel de la texture.

De manière à accélérer cette recherche et afin de minimiser les artefacts

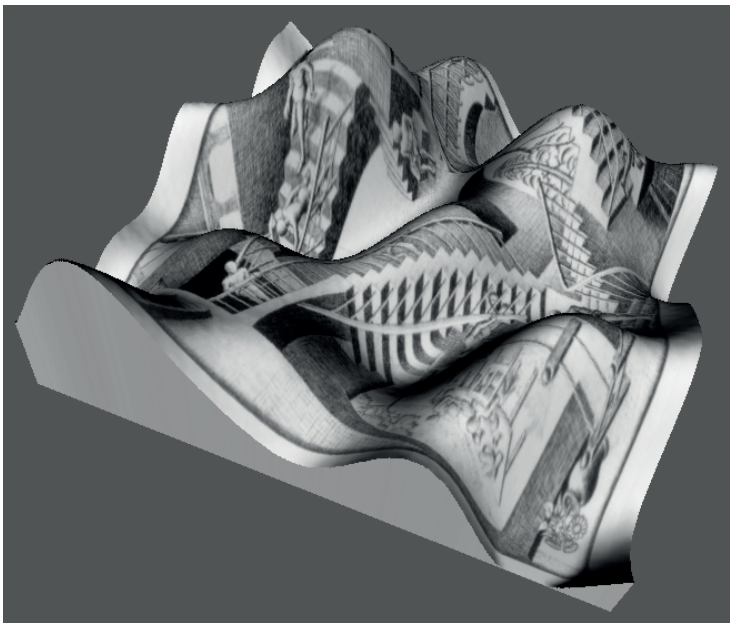


FIGURE 4.11 – Le *relief mapping* permet de représenter interactivement des mésostructures définies par champs de hauteurs. *images de [9]*

de discrétisation inhérents à la représentation explicite de la texture 2D, de récentes extensions [48, 9, 149] proposent d'effectuer une recherche linéaire permettant de s'approcher de la première intersection et d'obtenir ainsi une première approximation de solution suivie d'une recherche binaire afin de raffiner cette approximation.

Il est à noter également que ces travaux introduisent la courbure de l'objet à texturer de manière à appliquer cette technique sur des objets complexes. Ces méthodes donnent des résultats convaincants et résolvent de nombreux problèmes, comme les effets d'auto-occlusion et de silhouette

des mésostructures définies à base d'un champ de hauteur, comme illustré sur la figure 4.11.

Toutefois, ces méthodes peuvent devenir ardues à mettre en œuvre pour certaines mésostructures. En effet, une représentation se basant sur un champ de hauteurs consiste à élever des éléments de mésostructure le long de la normale à une surface de référence. Il n'est pas possible avec une telle représentation de créer des mésostructures avec des propriétés complexes, par exemple des mésostructures avec des cavités ou des éléments déconnectés. Pour répondre à ces problèmes, qui concernent la gamme de mésostructure qu'il est possible de représenter, Policarpo et al. proposent dans leurs travaux [126] une extension permettant d'effectuer le rendu de mésostructures plus complexes. Le principe consiste à définir la mésostructure d'un objet en utilisant non pas un mais plusieurs champs de hauteurs, comme illustré par la figure 4.12, permettant ainsi le rendu de mésostructures plus complexes. Toutefois, il est nécessaire d'adapter le nombre de champs de hauteurs (4 dans l'exemple de la figure 4.12) à chaque mésostructure.

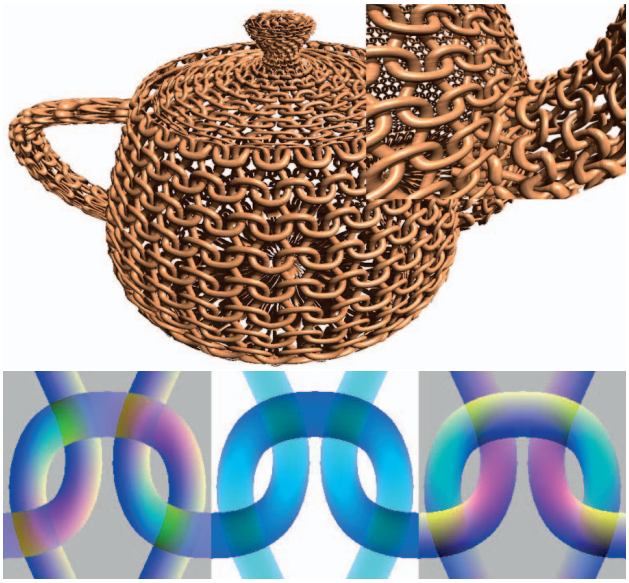


FIGURE 4.12 – Une représentation à l'aide de plusieurs champs de hauteurs permet de représenter des mésostructures plus complexes (images de [126]).

Le tableau 4.1 récapitule les avantages et inconvénients des principales méthodes de rendu exposées au cours de cette section. Dans ce tableau, nous

Méthode	Silhouette	Parallaxe	Qualité d'illumination	Diversité de mésostructure	Coût mémoire	Performances	Artefacts visuels
Bump Mapping[16]	Non	Non	-	-	++	++	
BTF [34]	Non	Oui	++	++	-	+	Ghosting
PTM [101]	Non	Non	+	+	+	++	Lissage
VDM [156]	Oui	Oui	-	-	-	+	Flou
Parallax Mapping [68]	Oui	Oui	-	-	++	++	Approximation
Relief Mapping [117]	Oui	Oui	+	-	++	+	

TABLE 4.1 – Récapitulatif des avantages et inconvénients des méthodes de cette section. Le symbole ++ (resp. -) indique une gestion réaliste (resp. restreinte) du paramètre correspondant.

avons attribué le symbole + (voir ++) aux méthodes offrant une gestion réaliste et de qualité du paramètre en question. Inversement, nous attribuons le symbole − aux techniques souffrant d'une gestion sommaire du paramètre considéré.

Premièrement, il est à noter que seules les méthodes procédant à une forme de reconstruction des informations de visibilité (comme les VDM et le *parallax mapping*) ou les méthodes de *relief mapping*, procédant à un calcul complet de la mésostructure, sont à même de rendre correctement la silhouette de la mésostructure. Les effets de parallaxes sont, quant à eux, globalement gérés par les méthodes présentées, à l'exception des méthodes faisant abstraction des différentes directions de vue, comme les PTM et le Bump Mapping.

Ce que nous appelons qualité d'illumination correspond ici au réalisme des effets d'illumination, comme la présence d'inter-réflexions ou d'ombres portées. Dans ce domaine, les méthodes présentant un net avantage sont les méthodes basées sur une mesure de l'apparence, comme les BTF et les PTM. Cette dernière souffre cependant d'un effet de « lissage » des interactions lumineuses complexes dû à la nature de sa représentation de mésostructure (un ensemble de coefficients pour retranscrire les interactions lumineuses).
De manière similaire et de par leur définition, les méthodes s'appliquant à une

large gamme de mésostructures sont celles basées sur une capture de l'apparence : les BTF et les PTM. De par leur nature, les méthodes procédant à une reconstruction de la mésostructure sont plus limitées. Ceci est principalement dû à une volonté d'accélérer la reconstruction de la mésostructure lors du rendu.

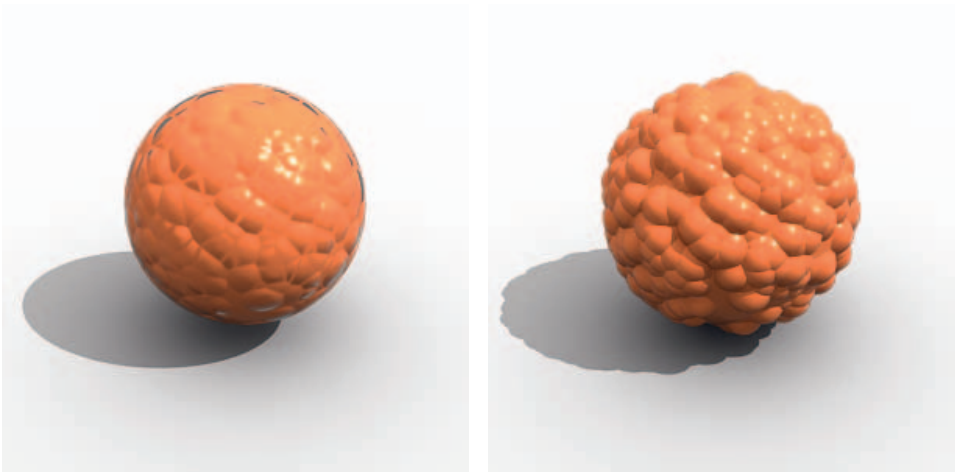
Le coût mémoire a une importance non négligeable pour une méthode interactive et montre la supériorité des méthodes procédant à une reconstruction de la mésostructure sur les méthodes basées image, comme les BTF et les VDM. Malgré l'emploi de méthodes de compression, le coût mémoire de ces méthodes reste cependant élevé.

Les performances des méthodes présentées ici sont globalement intéressantes. Les méthodes les plus simples, comme le *parallax mapping* et le *bump mapping* comptent parmi les plus rapides. La méthode des PTM est également intéressante pour la rapidité de reconstruction des effets lumineux. D'un autre côté, les BTF présentent des performances moindres. Ceci est notamment dû à la décompression des données qu'il est nécessaire d'effectuer lors du rendu (les données non compressées dépassent les capacités de stockage du matériel graphique). Les méthodes de *relief mapping* procèdent à une reconstruction, non pas par un calcul direct, mais par une recherche d'intersection. Cet algorithme de recherche, même s'il est optimisé, reste plus lent qu'une méthode procédant à un simple calcul, comme la méthode à base de PTM.

Au final, l'ensemble des méthodes vues dans cette section permettent le rendu de mésostructure en s'appuyant sur un calcul ayant lieu dans l'espace 2D correspondant à la surface du modèle. Nous allons voir dans la section suivante les approches qui considèrent la mésostructure dans l'espace de définition de l'objet lui-même.

4.1.2 Approche volumique

Les approches décrites dans cette section ajoutent des détails de mésostructure dans l'espace 3D du modèle en lui-même. Une des techniques d'ajout de détails les plus utilisées est celle du *displacement mapping*[30][31]. Cette technique crée effectivement des détails géométriques à la surface de l'objet. En se basant sur une représentation polygonale, le principe est de subdiviser la macro-structure à la surface de l'objet et de déplacer les sommets des polygones résultant le long de la normale à la surface, créant ainsi une mésostructure définie de manière géométrique à la surface de l'objet. L'utilisation d'une texture de champs de hauteurs permet de définir pour chaque sommet nouvellement créé son déplacement le long de la normale à la surface de l'objet originel. Cette technique donne des résultats intéressants et permet de résoudre naturellement les problèmes de parallaxe et de silhouette, comme présenté sur la figure 4.13.



(a) Bump mapping (b) Displacement mapping

FIGURE 4.13 – Les méthodes de *displacement mapping* permettent de résoudre des problèmes de parallaxe et de silhouette en procédant à un déplacement des sommets le long des normales.

Cette méthode pose néanmoins un problème important de par le nombre de polygones créés pour représenter la mésostructure. En effet, cette complexité est telle qu'elle rend l'application de cette méthode délicate pour des applications interactives. Quelques méthodes ont été proposées pour réduire ce problème. Par exemple, la génération des triangles directement sur le matériel graphique, proposée par Kryachko [79] et Asirvatham et al. [7] permet d'utiliser les

possibilités du matériel graphique pour obtenir des performances interactives. Livny et al. [97, 96] proposent dans la même optique la représentation d'un modèle à géométrie complexe par un découplage en un modèle grossier et un ensemble de champs de hauteurs (pouvant être considérés ici comme des informations de mésostructure). Ces méthodes, tirant parti des dernières améliorations graphiques, permettent d'ajouter rapidement des détails géométriques à un objet (voir figure 4.14), au prix toutefois d'un coût mémoire important. De plus, les effets complexes d'illumination, comme par exemple l'auto-ombrage des éléments de la mésostructure restent peu aisés à calculer. À l'instar des techniques reposant sur une carte de hauteurs, la gamme de mésostructures qu'il est possible de représenter avec cette méthode reste limitée.

(a) (b) (c)

FIGURE 4.14 – La méthode des *displacement patches*. Le modèle 3D Lucy (116 Millions de triangles).(a) représentation grossière. (b) Découpage du modèle grossier en patches. (c) Le modèle raffiné (64 fps). (images de [96])

Les techniques basées sur le *displacement mapping* souffrent toutes d'une gamme de mésostructures limitée. En effet, ces méthodes utilisent une carte de hauteurs pour définir la mésostructure. Cette représentation ne permet pas de modéliser des effets complexes, comme par exemple des éléments « surplombants » la surface, voire totalement déconnectés de l'objet. D'autres approches ont été proposées pour résoudre ce problème. Il en existe essentiellement deux catégories : les approches se basant sur une déformation totale et généralisée d'un volume et les approches ayant recours à un placage d'une couche épaisse sur la macrostructure d'un objet.

L'idée introduite par Perlin et Hoffert [123] s'inscrit dans la première catégorie : c'est le principe des *hypertextures*. Le principe des hypertextures est de considérer un objet de manière volumique, assignant une information de densité en chaque point de son volume. L'idée est de perturber cette information de densité en utilisant une méthode de déformation spatiale à base de turbulence. Ceci permet d'ajouter des détails de mésostructure volumique procédurale à un objet défini de manière volumique ou implicite, comme représenté sur la figure 4.15. Une étude plus détaillée des hypertextures a lieu dans le chapitre 7. Cette méthode est toutefois peu aisée à contrôler et impose la définition volumique ou implicite d'un objet.
La deuxième catégorie consiste à plaquer des parallélepipèdes rectangles sur la surface du modèle, créant ainsi une « surcouche » de mésostructure à l'objet [65] (voir la figure 4.16). Avec les avancées du matériel graphique, cette approche a été raffinée et offre désormais des performances interactives. Par exemple, des méthodes [106] [66] représentent les mésostructures volumiques sous la forme d'un ensemble de polygones semi-transparents texturés. Ces polygones sont toujours placés en extrusion de la surface et permettent le rendu de la mésostructure en utilisant ces polygones comme géométrie de substitution, à l'instar des techniques de rendu volumique à base de *slicing* (voir section 3.3.2). Ces méthodes offrent de bons résultats, mais peuvent causer des artefacts lorsque les angles de vue sont rasants.
Afin de résoudre ces problèmes, plusieurs travaux [88] [119] génèrent pour chaque polygone d'un objet un prisme extrudé et calculent l'intersection de ces prismes avec un ensemble de plans perpendiculaires à la direction de vue. Cette intersection donne la géométrie de substitution qui sera utilisée en conjonction avec une texture 3D pour effectuer le rendu à base de *slicing* de la mésostructure volumique.

Plus récemment, Decaudin et al. [36] proposent d'appliquer cette approche

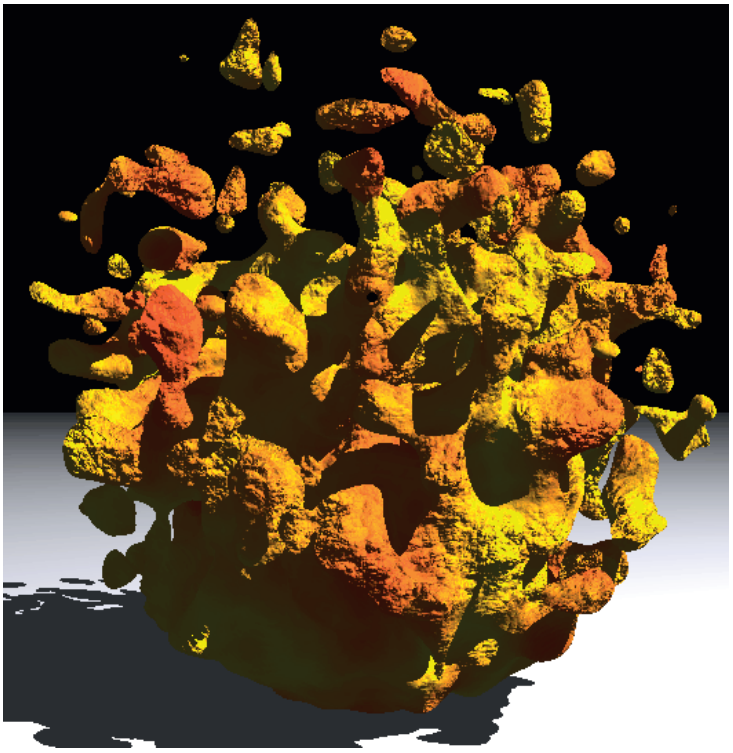


FIGURE 4.15 – Les hypertextures permettent d'ajouter des détails de mésostructure volumiques procéduraux à un objet.

au principe des *billboards*[2]. L'idée des *billboards volumiques* est proche des méthodes de *slicing* (section 3.3.2). Le principe est de définir un ensemble de prismes, qui serviront de géométrie de substitution, chacun étant associé à une texture 3D. Lors du rendu, un ensemble de tranches parallèles au plan image est généré par le matériel graphique. L'intersection des tranches avec les prismes permet de créer un ensemble de poylgones. Ces polygones sont texturés (en utilisant un mécanisme de coordonnées de texture similaire au slicing) et leur composition restitue l'apparence de l'objet volumique à représenter. Cette méthode permet également d'appliquer une mésostructure à la surface d'un objet (voir figure 4.17). L'ensemble de ces méthodes reste toutefois contraintes par leur représentation explicite (sous forme d'une texture 3D).

Afin d'améliorer la gamme de mésostructure qu'il est possible de représenter, Wang et al. [157] proposent une extension des VDM permettant de trai-

2. Un billboard est utilisé comme imposteur pour représenter un objet complexe par un ou plusieurs plans semi-transparents texturés.

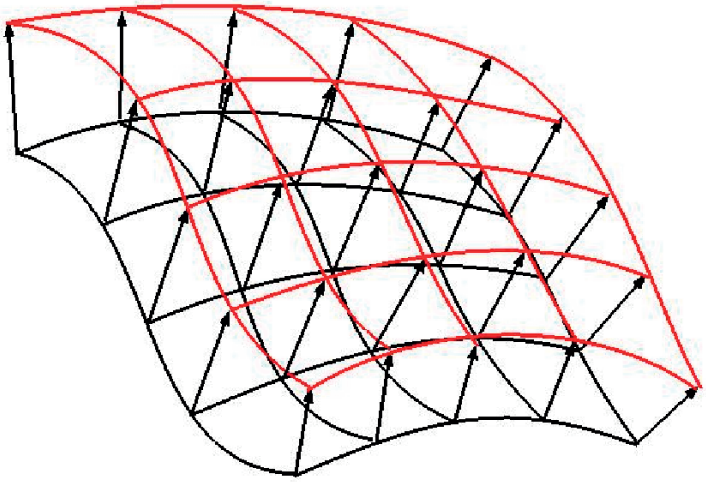


FIGURE 4.16 – Le plaquage de texture volumique permet de créer une « couche » à la surface de l'objet représentant la mésostructure.

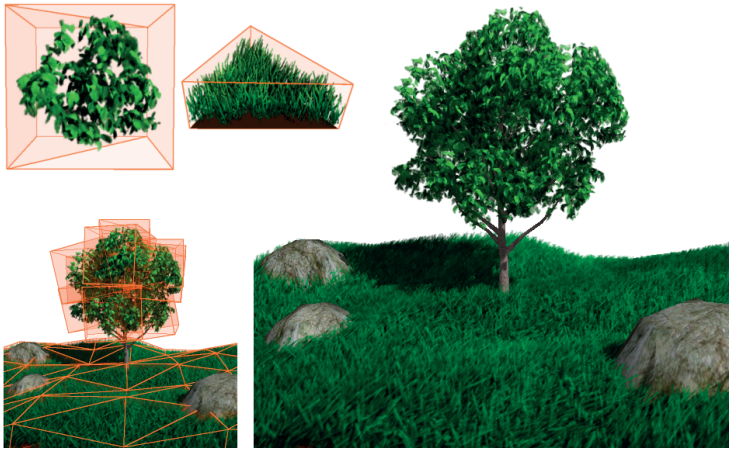


FIGURE 4.17 – Les *billboards* volumiques permettent un rendu réaliste et rapide d'objets volumiques complexes.Images de [36]

ter les mésostructures volumiques : les *Generalized Displacement Maps*(GDM). Cette extension permet de résoudre des problèmes inhérents au VDM en permettant la représentation de structures non-basées sur un champ de hauteurs. Le principe de cette méthode est identique aux VDM mais utilisé dans un contexte volumique (c'est-à-dire qu'une information de distance à l'élément solide de la mésostructure la plus proche dans un ensemble de directions arbitraires est stockée en chaque point d'un volume 3D). Le rendu est effectué en extru-

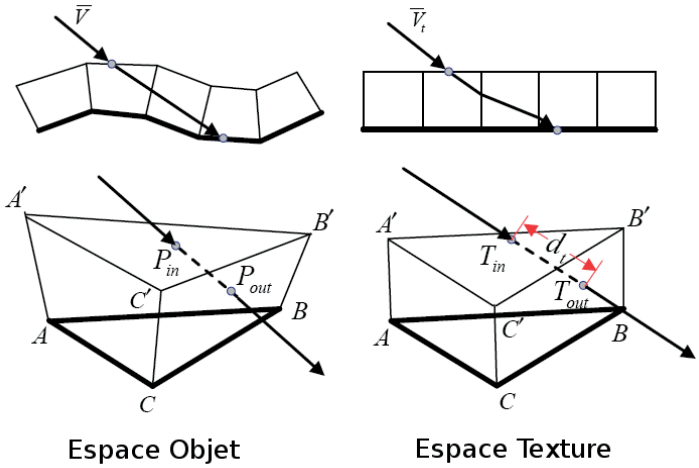


FIGURE 4.18 – Principe des *generalized displacement maps*. L'extrusion des polygones de la surface de l'objet permet de reconstruire la trajectoire du rayon de vue dans l'espace 3D de la mésostructure. Image de [157]

dant les polygones de la surface de base et en utilisant cette information de distance pour chaque rayon de vue. Le principe de la méthode est résumé par la figure 4.18. Cette méthode est efficace, interactive et couvre une large gamme de mésostructures, comme illustré par la figure 4.19. Toutefois, la représentation utilisée par cette méthode reste une fonction 5D discrète et pose le problème à la fois de l'espace mémoire et de la résolution de la mésostructure représentée, ce qui limite son application à un petit échantillon répété sur la surface d'un modèle.

Un des problèmes complexes du rendu de mésostructures encore peu abordé dans cette section est le problème des interactions lumineuses, difficiles à intégrer dans le rendu même de la mésostructure. La problématique de l'évaluation des interactions lumineuses complexes dans les mésostructures volumiques est approchée par Chen et al. [26] dans les *Shell Textures Functions* (STF). L'idée, représentée sur la figure 4.20 consiste à définir une mince couche sous la surface et à calculer par méthode de lancer de photons la diffusion de la lumière sous la surface de l'objet. Comme le montre la figure 4.20, les résultats sont impressionnants mais les performances de cette méthode ne sont pas suffisantes pour un rendu interactif. Des extensions plus récentes améliorent ce point, telle la méthode des *Shell Radiance Textures Functions* [141] qui permet le rendu en temps interactif mais ne parvient plus de ce fait à rendre correctement les effets de silhouette de la mésostructure pour des angles de vue rasants.

FIGURE 4.19 – Un exemple de GDM avec illumination globale.Image de [157]

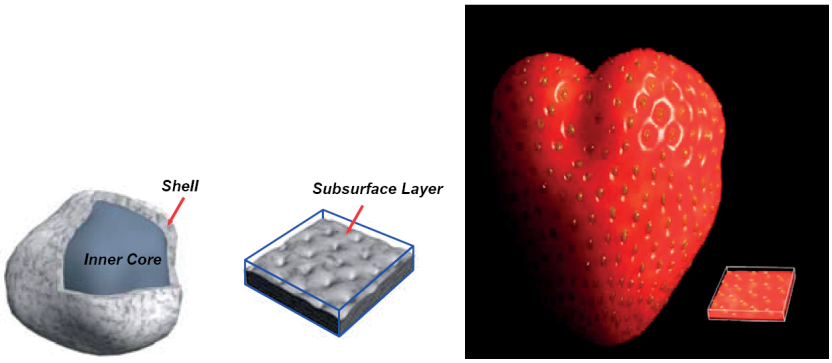


FIGURE 4.20 – *Shell Textures Function* [26]. À gauche : les objets sont définis comme une mince couche hétérogène autour d'un cœur opaque. À droite : une mésostructure semitransparente appliquée à un modèle de fraise. Images de [26]

Le tableau 4.2 présente un récapitulatif des avantages et inconvénients des différentes approches volumiques traitées dans cette section. On peut remarquer que les effets de silhouette et de parallaxe sont traités par toutes les méthodes, à l'exception de la méthode des *Shell radiance texture function*, qui ne gère pas la silhouette. La meilleure qualité d'illumination est sans conteste obtenue avec la méthodes des STF. Cette méthode procède en effet à une évaluation des effets lumineux complexes, au prix toutefois d'un temps de calcul dépassant de plusieurs ordres de grandeur les autres méthodes.

D'une manière globale, les méthode présentées dans cette section s'appliquent à une plus grande gamme de mésostructures, à l'exception toutefois du *displacement mapping*, basé sur une carte de hauteurs.
En ce qui concerne le coût mémoire, la méthode la plus efficace est celle du *displacement mapping*. La technique des GDM , malgré sa technique de compression, reste cependant plus coûteuse en terme de mémoire que les méthodes basées par exmple sur une texture 3D, comme dans le cas des billboards volumiques et des méthodes de slicing.
L'ensemble des techniques basées sur une extrusion de la surface ou sur une texture 3D plaquée sur la surface est sujet à des problèmes de paramétrisation et à des artefacts de déformation, voire même d'auto-intersection, dans le cas de surfaces à forte courbure. Toutefois, les performances des méthodes récentes, tels les *billboards* volumiques, de par l'absence d'une phase de tri des primitives et une implémentation efficace sur le matériel graphique, permettent un rendu rapide des mésostructures. Il est à noter que les méthodes de displacement mapping ont une efficacité dépendant du nombre de polygones générés.

On peut remarquer qu'une des techniques donnant les résultats visuels les plus proches de la réalité est la méthode des STF, au prix toutefois d'un temps de calcul prohibant une utilisation interactive.

Méthode	Silhouette	Parallaxe	Qualité d'illumination	Diversité de mésostructure	Coût mémoire	Performances	Artefacts
Displacement Mapping [30]	Oui	Oui	-	-	+		
Slicing [119]	Oui	Oui	-	++	+	+	Illumination
Billboards Volumiques [36]	Oui	Oui	+	++	-	++	
GDM [157]	Oui	Oui	+	+	++		
STF [26]	Oui	Oui	++	++	-	-	
SRTF [141]	Non	Oui	+	+	-	-	

TABLE 4.2 – Récapitulatif des avantages et inconvénients des méthodes de cette section. Le symbole ++ (resp. -) indique une gestion réaliste (resp. restreinte) du paramètre correspondant.

Chapitre 5

Rendu de mésostructure par lancer de rayon interactif

Nous avons vu au cours du chapitre précédent les différentes méthodes permettant le rendu de mésostructure d'un modèle. Nous avons également pu préciser les phénomènes participant au réalisme des images de synthèses générées. Nous allons dans ce chapitre étudier la méthode de rendu de mésostructures volumiques proposée dans ces travaux de thèse dont les détails sont décrits dans la section 5.1.
Ces travaux se basent sur une méthode hybride. Alors que les méthodes de rastérisation et de lancer de rayon présentent chacunes des avantages et inconvénients (3), le concept d'une méthode hybride est d'allier les avantages de ces deux méthodes, afin d'obtenir une visualisation riche visuellement sans perdre des performances. La méthode hybride que nous proposons repose sur l'utilisation conjointe des principes de rasterisation et de lancer de rayons. Il est à noter qu'alors qu'à ce jour le concept d'une approche hybride est utilisée dans de nombreux travaux [12], il n'en était pas de même lors du début de ces travaux.

Les différentes applications de cette méthode de rendu présentées dans les chapitres suivants seront l'occasion de discuter des différents phénomènes que cette méthode permet de représenter. Finalement, nous concluons ce chapitre (section 5.2) en discutant des avantages et inconvénients inhérents à notre méthode hybride.

5.1 Principe général de notre méthode hybride

La méthode que nous proposons dans ces travaux de thèse présente quelques similarités avec certaines des méthodes vues dans le chapitre précédent, en particulier la méthode du *relief mapping*. Afin de s'affranchir des problèmes

de discrétisation et de stockage des méthodes basées sur un précalcul, nous souhaitons également utiliser la puissance de calcul du GPU afin de procéder à la reconstruction de la mésostructure lors du rendu. À l'instar de ces méthodes, notre méthode se repose sur une étape de rasterisation de la macrostructure de l'objet pour générer un ensemble de rayons de vue (un pour chaque pixel). Chaque rayon permet ensuite l'évaluation de la contribution de la mésostructure pour le pixel associé.

Cependant, notre méthode présente quelques différences par rapport aux méthode citées précédemment. Premièrement, contrairement aux méthodes de *relief mapping*, notre approche va être principalement de type volumique, c'est-à-dire que nous restons dans l'espace de l'objet et non pas dans l'espace paramétré à la surface de l'objet. Cela nous permet de nous abstraire du problème de paramétrisation des surfaces et de pouvoir appliquer cette technique aux modèles définis de manière volumique. Deuxièmement, notre technique ne se base pas sur le calcul de l'intersection d'un rayon de vue avec la mésostructure mais sur l'ajout de détails sur l'ensemble du rayon par méthode de *ray marching*. Ceci présente l'avantage de permettre le rendu de mésostructure définie de manière volumique, et élargit la gamme de mésostructure auxqelles cette méthode peut s'appliquer. L'avantage d'une méthode de *ray marching* par rapport à une méthode à base de *slicing*, comme les *billboards volumiques*, est principalement de pouvoir définir un schéma d'échantillonnage pour chaque rayon. Alors qu'il est difficile de contrôler précisemment le schéma d'échantillonnage (non consistant pour chaque rayon, voir section 3.3.2) avec ces méthodes, notre technique offre une souplesse dans la définition du schéma d'échantillonnage. De plus, notre technique permet de s'abstraire totalement de la macrostructure.

Ce que vise à réaliser notre méthode de rendu est au final d'ajouter des détails de mésostructure le long de l'intersection entre un rayon de vue et l'objet. Cela revient, comme l'illustre la figure 5.1, à calculer la contribution de la mésostructure le long du segment $[pe_i, ps_i]$ pour chaque rayon de vue issu d'un pixel f_i. Nous faisons ici l'hypothèse que la mésostructure ne doit apporter aucune contribution avant le point pe_i ou après le point ps_i. Il est donc nécéssaire de calculer ces deux points délimitant la contribution de la mésostructure à l'objet. Il n'est pas envisageable de se contenter de calculer uniquement le point d'entrée du rayon de vue dans l'objet (pe_i) sans affecter la généralité de notre méthode. En effet, la contribution de la mesostructure le long du segment peut être nulle (transparente) le long du segment $[pe_i, ps_i]$ mais rien ne garantit qu'elle sera également nulle sur $[ps_i, \infty[$. Cela dépend fortement de la méthode d'évaluation de la mésostructure. Afin de garantir un résultat correct,

il convient donc de calculer le point de sortie du rayon de vue de l'objet (ps_i)à afin de ne pas rajouter de détails de mésostructure n'appartenant pas à l'objet.

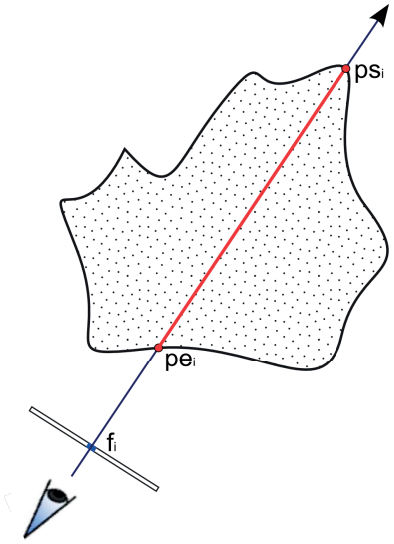


FIGURE 5.1 – L'intersection d'un rayon de vue issu du fragment f_i peut s'exprimer pour un objet convexe comme le segment $[pe_i, ps_i]$

Nous considérons dans ces travaux que l'objet à texturer est représenté sous la forme d'une modélisation discrète, notamment, un nuage de point. Nous utilisons principalement un ensemble de splats afin d'obtenir une définition simple et permettant une reconstruction de l'objet basée sur un filtre de reconstruction. Cet objet est affiché en utilisant une méthode traditionnelle à base de rasterisation. Au cours de cette phase, l'idée est, à l'instar des méthodes présentées précédemment, non pas d'obtenir une information de couleur, mais des informations permettant le calcul des points pe_i et ps_i.

Par la suite, nous allons distinguer deux types d'objets : Les objets définis par une information de densité en chaque point du volume et les objets définis par leur frontière uniquement.

Il est à noter que l'affichage de ces deux représentations est réalisé de manière trés différente. En effet, dans le cas de modèles surfaciques, il s'agira simplement d'utiliser les techniques de rendu traditionnelles (voir section 3.3.1). Les objets volumiques quant à eux requièrent une technique de rendu différente (basée dans notre cas sur une méthode de *splatting* comme présenté à la section 3.3.2). Il est donc nécéssaire de faire la distinction entre ces deux classes

d'objets dans la suite de ces travaux de thèse.

Nous commençons cette section en présentant notre méthode de rendu pour les objets définis par leur bord dans la section 5.1.1. La section 5.1.2 décrit quant à elle notre méthode de rendu volumique.

Indépendamment de la représentation choisie pour l'objet, nous nous retrouvons avec un segment $[pe_i, ps_i]$ le long duquel il est nécéssaire d'ajouter les détails de mésostructure. Cette phase d'ajout de mésostructure est indépendante des autres phases et est fortement correlée à la mésostructure en elle-même. Nous présenterons cette phase d'ajout de détails dans la section 5.1.3.

5.1.1 Objet surfacique

Nous nous concentrons ici sur un objet défini de manière surfacique par son bord. Nous supposons dans ce chapitre que la représentation est basée sur un ensemble de primitives orientées, comme des splats et les triangles. Pour simplifier, nous considérons dans l'immédiat que l'objet que nous souhaitons afficher est convexe. Nous présenterons par la suite les techniques permettant de traiter un objet non-convexe.
Le but de cette étape étant de récupérer pour chaque fragment le segment $[pe_i, ps_i]$, il nous faut obtenir les coordonnées du point d'entrée (respectivement de sortie) pe_i (resp. ps_i) du rayon de vue issu du fragment f_i correspondant. Nous nous basons dans cette section sur les techniques de rasterisation de primitives actuelles.

Pour ce faire nous effectuons un rendu en deux passes, à l'instar de [77]. Le principe est d'utiliser au cours de la rasterisation le tampon de profondeur de la carte graphique pour récupérer pour chaque pixel la profondeur des intersections entrantes et sortantes du rayon de vue et de l'objet considéré. En pratique, l'idée est d'effectuer deux passes : Une première permettant de récupérer la profondeur de l'intersection de sortie (la profondeur à laquelle le rayon de vue quitte l'objet) et une autre permettant de récupérer la profondeur de l'intersection d'entrée dans le modèle, en utilisant l'information de profondeur fournie par le Z-buffer. Pour une représentation basée sur des primitives orientées, il est aisé de différencier si l'intersection, entre le rayon de vue et la surface d'un objet, représente l'entrée ou la sortie du rayon de vue dans l'objet. Il suffit de déterminer si l'orientation fait face à l'utilisateur ou non.

La première étape est de rasteriser les primitives « arrières » du modèle et de stocker dans une texture intermédiaire la valeur de profondeur donnée par le tampon de profondeur. En utilisant l'interpolation de la carte graphique, ceci

permet d'obtenir en chaque pixel les coordonnées dans l'espace objet de la surface (arrière) se projetant sur ce pixel, c.-à-d. les coordonnées de l'extrémité du segment de vue ps_i. La seconde passe de cette méthode affiche quant à elle les primitives avant de l'objet et permet de la même manière d'obtenir les coordonnées du point d'entrée pe_i du rayon de vue dans l'objet. En récupérant les informations obtenues lors de la première passe, nous obtenons pour chaque fragment le segment défini dans l'espace objet qui correspond à l'intersection entre le rayon de vue et le modèle.

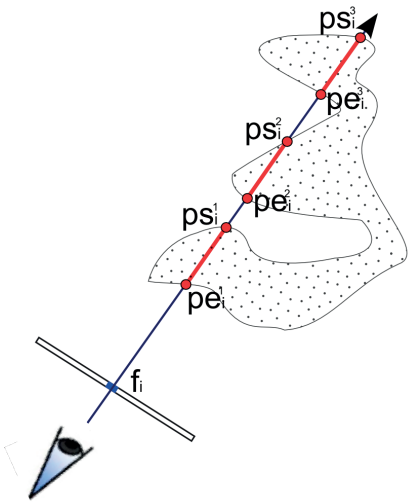


FIGURE 5.2 – L'intersection d'un rayon de vue et d'un objet non-convexe peut être composée de plusieurs segments disjoints.

Cette méthode est efficace et permet d'obtenir rapidement le segment $[pe_i, ps_i]$ dans l'espace objet. Néanmoins, cette méthode devient plus complexe dans le cas d'objets non convexes. En effet, comme illustré par la figure 5.2, un rayon de vue issu d'un fragment f_i de l'écran peut avoir plusieurs intersections distinctes avec l'objet (et donc plusieurs segments $[pe_i, ps_i]$). Choisir seulement la première intersection (le segment pe_i^1, ps_i^1 sur la figure 5.2) peut conduire à de sévères artefacts visuels liés à la mésostructure. En effet,la mésostructure peut être transparente pour le segment pe_i^1, ps_i^1 mais opaque pour les segments suivants. Ne considérer que la première solution peut conduire aux artefacts présentés sur la figure 5.3. Une solution correcte consiste à trier les primitives en fonction de leur distance à l'écran avant de les envoyer à la carte graphique et de composer les différents segments dans le bon ordre. Cette solution, bien que correcte visuellement, est en pratique irréalisable en temps interactif pour des

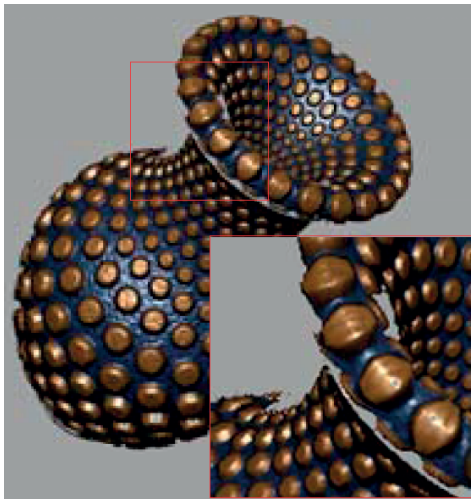


FIGURE 5.3 – Des artefacts de visibilité peuvent apparaître lors d'angles de vue rasant lorsqu'on ne traite que la première intersection. *Image de [157]*

modèles composés de nombreuses primitives car elle ne tire dans ce cas pas parti des capacités de parallélisation du matériel graphique.

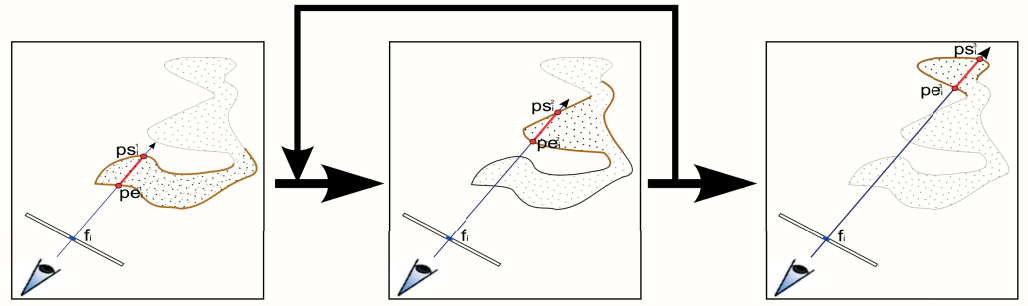


FIGURE 5.4 – Le principe du *Depth Peeling* consiste à traiter séquentiellement les intersections d'un rayon de vue avec l'objet.

La technique de *Depth Peeling*, proposée par [102], permet de résoudre ce problème de manière efficace. L'idée de cette méthode est de traiter ces différentes intersections le long du rayon de vue de manière séquentielle en utilisant plusieurs passes de rendu (figure 5.4). Pour chaque passe, toutes les primitives du modèle sont projetées sur l'écran. Le principe est de récupérer au cours de la $n^{ème}$ passe les informations nécéssaires pour reconstruire le segment pe_i^n, ps_i^n, à savoir la profondeur de la $n^{ème}$ entrée (et $n^{ème}$sortie) du rayon de vue dans

l'objet. Pour pouvoir conserver uniquement cette $n^{ème}$ paire d'intersections, nous utilisons le mécanisme du tampon de profondeur de la carte en conjonction avec un second buffer artificiel de profondeur. L'idée est de stocker dans ce nouveau buffer pour chaque pixel la profondeur de la dernière intersection traitée. De cette manière, il suffit au cours de la $n^{ème}$ passe d'écarter les fragments ayant une profondeur inférieure à la dernière profondeur traitée (on considère que ces fragments ont déjà été traités au cours des $n-1$ passes précédentes). Le mécanisme standard de tampon de profondeur d'OpenGL nous permet alors de récupérer les premières intersections parmi les fragments restants. Nous obtenons effectivement la $n^{ème}$ paire d'intersections. Le pseudo-code de l'algorithme 5.1 résume ce procédé. Pour chaque segment ainsi reconstruit, des détails de mésostructure sont ajoutés (voir section suivante). La contribution de ce segment au fragment final est combinée avec les contributions des segments précédents (stockées dans un buffer intermédiaire). Ceci permet une reconstitution séquentielle correcte des segments $[pe_i^n, ps_i^n]$ pour le fragment f_i.

Il existe plusieurs méthodes pour déterminer l'arrêt de cette méthode. Intuitivement, cette technique doit se terminer lorsque toutes les intersections de tous les rayons de vue avec l'objet ont été traitées. Ceci peut se determiner en utilisant le mécanisme d'*occlusion query* d'OpenGL. En pratique, ce mécanisme permet de compter le nombre de fragment générés par une passe. Ceci permet dans notre cas de connaitre le nombre de fragments passant les tests de profondeur décrit précédemment (c'est à dire les fragments correspondant effectivement à une nouvelle intersection). De manière intuitive, si le nombre de fragments générés est nul, cela signifie qu'il n'y a plus de segments à traiter et que l'algorithme peut donc s'arrêter. Toutefois, ce mécanisme comporte un coût en terme de performance. Une approximation courante consiste à fixer à l'avance le nombre de passes k à effectuer. Ceci permet parfois d'augmenter les performances tout en réduisant la qualité visuelle (seuls les k premiers segments sont considérés, les autres étant ignorés).

Cette méthode, bien que plus lente qu'une méthode considérant uniquement la première intersection, permet de traiter rapidement les objets non-convexes et reste suffisante pour notre application. Cette méthode est actuellement multipasse mais pourrait être résumée en une seule passe en considérant de futures améliorations du matériel graphique, tel le support matériel des *k-buffers* ([13]), ce qui améliorerait probablement les performances.

Algorithme 5.1 Pseudo-code de la méthode de rendu de macrostructure utilisée. La technique de *Depth Peeling* [102] utilisée permet de traiter les objects non-convexes.

pour tout passes p **faire**
  Projection des primitives dans l'espace écran
  pour tout fragments f générés **faire**
    Comparaison et défausse des fragments ayant une profondeur inférieure à la valeur stockée dans le tampon intermédiaire
    Sélection du fragment d'**entrée** dans l'objet ayant la profondeur la plus faible (pe_i^p) grâce au Z-buffer
  fin pour
  Projection des primitives dans l'espace écran
  pour tout fragments f générés **faire**
    Comparaison et défausse des fragments ayant une profondeur inférieure à la valeur stockée dans le tampon intermédiaire
    Sélection du fragment de **sortie** dans l'objet ayant la profondeur la plus faible (ps_i^p) grâce au Z-buffer
    Evaluation du segment $[pe_i^p, ps_i^p]$ et combinaison de sa contribution dans un buffer intermédiaire
    Mise à jour du tampon intermédiaire (valeur de profondeur ps_i^p)
  fin pour
fin pour

5.1.2 Objet volumique

Dans cette section, nous abordons le problème du rendu d'objets définis de manière volumique. Pour le rendu, notre méthode se base sur l'EWA splatting ([25]) présentée dans la section 3.3.2. Nous considérons dans cette section que la macro-structure est modélisée à partir d'un nuage de points échantillonné irrégulièrement. Chacun de ces points est, à l'instar du splatting, associé à un noyau d'interpolation afin de permettre une reconstruction continue de l'objet. L'idée principale de cette méthode consiste à reconstruire l'apparence finale d'un objet grâce à la combinaison des projections de ces noyaux d'interpolation dans l'espace écran.

Nous considérons dans cette section que les noyaux de reconstruction sont des gaussiennes 3D radialement symétriques tronquées (cette hypothèse est couramment utilisée dans les algorithme de *splatting*). L'avantage principal d'un tel noyau réside dans sa projection en l'espace écran. En effet, la projection d'une telle gaussienne 3D vers le plan image donne comme image une

gaussienne 2D fermée, ce qui facilite l'évaluation de la contribution du noyau de reconstruction.

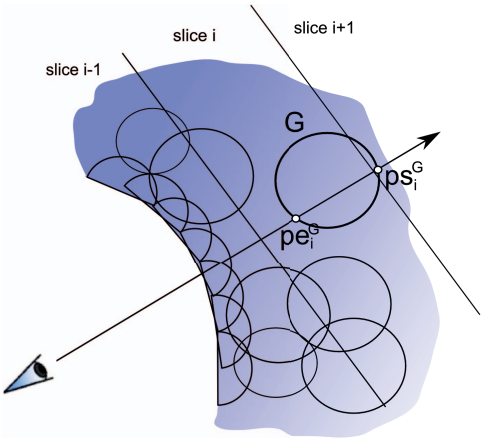


FIGURE 5.5 – Notre méthode d'ajout de détails pour des macrostructures volumiques considère l'objet comme une collection de noyaux de reconstruction. Nous souhaitons ici rajouter des détails le long du segment défini par l'intersection d'un rayon de vue avec chaque noyau de reconstruction G.

La figure 5.5 présente une vue d'ensemble de notre méthode de rendu volumique. Nous ne considérons pas dans ce cas l'intersection des rayons de vue avec le "bord" de l'objet. Nous allons considérer l'intersection des rayons de vue avec chaque noyau de reconstruction G et ajouter des détails à chaque segment $[pe_i^G, ps_i^G]$ ainsi défini.

A l'instar du splatting, les points sont dans un premier temps triés et regroupés en tranches parallèles au plan de la caméra. Ce tri est effectué sur le CPU et les indices des points ainsi triés sont transmis à la carte graphique [1]. Les tranches sont traitées séquentiellement de la plus proche à la plus lointaine. L'image représentant la contribution d'une tranche du modèle dans l'image finale est décrite comme la somme pondérée de la projection dans l'espace écran des noyaux de reconstruction associés à chaque point. Dans un premier temps, les noyaux de reconstruction sont projetés sur l'écran, donnant ainsi pour chaque noyau une *empreinte 2D*. Techniquement, cela revient à représenter chaque noyau par une primitive *OpenGL point*. Ces points sont projetés sur l'écran et rasterisés sous la forme de carré dans l'espace écran. Lors de cette

1. Les indices des points sont envoyés à la carte (et non les points eux mêmes, de manière à limiter les transferts du CPU au GPU) en utilisant le mécanisme des *Vertex Buffers Objects* d'OpenGL.

étape, le vertex program estime de manière conservatrice la taille du splat projeté en se basant sur une division perspective du rayon le plus large de l'ellipse par la profondeur dans l'espace de la caméra du centre du splat. Ceci permet de rasteriser un sommet OpenGL comme un carré de taille $d_i \times d_i$ dans l'espace image. Chaque fragment de ce carré est ensuite testé par le fragment program pour déterminer si il se trouve à l'intérieur ou à l'extérieur du contour elliptique projeté, c'est-à-dire s'il doit être défaussé (il n'appartient pas à la projection) ou traité.

Il convient ensuite de déterminer quelle sera l'information générée pour chaque fragment appartenant à une empreinte. Dans notre cas, cela se résume à générer les segments $[pe_i^G, ps_i^G]$. Premièrement pour générer ces segments, nous prenons comme point de départ le fait que les noyaux de reconstruction G (des Gaussiennes 3D radialement symétriques tronquées) peuvent être englobées par une sphère. De ce fait, nous pouvons faire l'hypothèse suivante : *un rayon partant du centre de projection de la caméra et passant par un fragment généré par la rasterisation d un noyau de reconstruction G intersecte la sphère englobante S de ce noyau.* Il reste à déterminer alors dans le fragment program les coordonnées pe et ps dans l'espace objet des points d'intersection du rayon de vue avec la sphère S. Ce segment représente la contribution du noyau de reconstruction G pour le fragment f_i. La contribution du noyau G résultant sera pondérée (voir 3.3.2) et additionnée dans un buffer intermédiaire.

Une fois que tous les noyaux d'une tranche du modèle ont été projetés, l'étape de normalisation des contributions permet d'obtenir la contribution de cette tranche pour l'image finale. Cette contribution est alors combinée aux contributions des tranches prédédentes de manière traditionnelle (voir section 3.3.2).

Il est à noter que dans cette méthode, chaque fragment reçoit les contributions de plusieurs noyaux pour une même zone de l'espace objet, comme souligné par la figure 5.6. Ce phénomène, bien que coûteux en temps de calcul, apporte néanmoins un antialiassage naturel aux images produites. En effet, pour un même rayon de vue, les contributions de plusieurs segments $[pe_i^G, ps_i^G]$, $[pe_i^{G'}, ps_i^{G'}]$,... sont mélangées. Ceci introduit une erreur de visibilité communes aux algorithmes de *splatting* (voir section 3.3.2). Toutefois, il convient de noter que ces segments, même s'ils se recouvrent, sont rarement identiques et de ce fait sont échantillonnés de manière différente. Ceci provoque un suréchantillonnage dense de la région couverte par ces noyaux de reconstruction, ce qui permet souvent d'éviter les effets de sous-échantillonnage communs aux

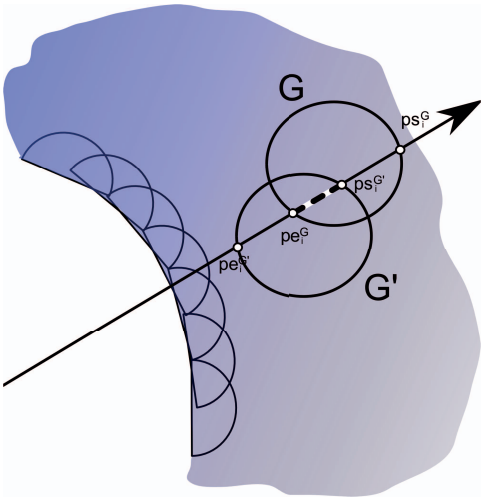


FIGURE 5.6 – Les noyaux G se recouvrant, une erreur de visibilité commune aux algorithme de splatting est introduite. En effet, un segment le long d'un rayon de vue est recouvert par plusieurs noyau et donc est ainsi traité plusieurs fois (en vert sur cette figure).

techniques de lancer de rayon. Ceci introduit dans notre méthode un phénomène d'anti-aliassage naturel. Toutefois, ce sur-échantillonnage a un impact non négligeable sur les performances de la méthode.

5.1.3 Rendu de mésostructure avec un lancer de rayon GPU

Après le passage de chaque primitive par l'étape de rasterisation, nous obtenons pour chaque fragment f_i une liste ordonnée de segments $[pe_i, ps_i]$ correspondants dans l'espace objet. C'est le long de ces segments que seront ajoutées les informations de mésostructure. Les calculs effectués au cours de cette phase dépendent de la manière dont la mésostructure est représentée. Cette même approche est valable pour les deux représentations : surfacique ou volumique.

L'idée est de calculer l'apparence de la mésostructure par un mécanisme de *ray marching* « local », c'est-à-dire d'échantillonner le segment $[pe_i, ps_i]$ et d'évaluer la contribution de la mésostructure à chaque échantillon. Nous employons le terme « local » car cet échantillonnage est réalisé uniquement sur les segments $[pe_i, ps_i]$ et non sur l'ensemble de l'objet. Ces opérations sont effectuées par un *fragment program* pour chaque fragment généré par l'étape de rasterisation. L'algorithmes 5.2 présente les opérations effectuées dans le cas du rendu d'un objet surfacique. Il s'agit de générer un ensemble d'échantillons le long du

Algorithme 5.2 Algorithme de *ray marching* local pour un objet surfacique.

p_e : Coordonnées d'entrée du rayon de vue
p_s : Coordonnées de sortie du rayon de vue
p_s : Coordonnées de sortie du rayon de vue
sampling : Distance entre deux échantillons successifs
couleur : Couleur finale
opacité : Opacité finale
nb : Nombre d'échantillons
pos_e : Position de l'échantillon courant
col_e : Couleur de l'échantillon courant
opa_e : Opacité de l'échantillon courant
couleur $\leftarrow (0.0, 0.0, 0.0)$
opacité $\leftarrow 0.0$
nb $\leftarrow \min(\frac{(p_s - p_e)}{\text{sampling}}, 1)$
$\text{pos}_e \leftarrow p_e$
tantque opacité < 1.0 et nb > 0 **faire**
  $(\text{col}_e, \text{opa}_e) \leftarrow$ évaluation(pos_e)
  $(\text{col}_e) \leftarrow$ ombrage$(\text{pos}_e, \text{col}_e)$
  (couleur) $\leftarrow$ couleur $+ (1 -$ opacité$) * \text{col}_e$
  (opacité) $\leftarrow$ opacité $+ (1 -$ opacité$) * \text{opa}_e$
  $\text{pos}_e \leftarrow \text{pos}_e + \frac{p_s - p_e}{\text{nb}}$
  nb $\leftarrow$ nb $- 1$
fin tantque
retourner(couleur, opacité)

segment $[pe_i, ps_i]$ du fragment correspondant et d'utiliser une fonction d'évaluation pour calculer la contribution de la mésostructure.
La différence principale dans le cas d'un modèle volumique toutefois et qu'il est nécéssaire de reconstruire le segment $[pe_i, ps_i]$ à partir de l'intersection du rayon de vue et de la sphère englobante de G. Il est également à noter que la contribution de la mésostructure est ensuite pondérée par une gaussienne centrée sur la projection du centre de G (le filtre de reconstruction associé au splat). L'ensemble des contributions des fragments est accumulée et normalisée avant l'affichage final. L'évaluation de la contribution de l'échantillon est effectuée par la fonction $evaluation()$ et est dépendante de la représentation de la mésostructure. Les contributions des échantillons sont accumulées selon le schéma de composition classique du rendu volumique (équation 3.9). Le nombre d'échantillons généré pour chaque segment $[pe_i, ps_i]$ est une variable définie par l'utilisateur final et influe fortement sur la qualité et les performances de la méthode de rendu.

Plusieurs types de mésostructure peuvent être représentées avec notre principe de rendu par *ray marching* local. Nous présentons dans ce travail de thèse deux applications traitant chacune d'une représentation de mésostructure spécifique :

- La première application (chapitre 6) traite le cas de mésostructures définies de manière explicite et plus particulièrement de mésostructures stockées dans des textures 3D. Le schéma d'évaluation de la mésostructure consiste à « plonger » le rayon de vue dans la texture et à indexer une texture 3D avec les coordonnées de l'échantillon.
- La deuxième application (chapitre 7) concerne le cas de mésostructures définies de manière procédurale. La fonction d'évaluation du modèle consiste simplement en l'évaluation du modèle procédural définissant la mésostructure.

5.2 Évaluation et discussions

Dans la section précédente, nous avons présenté la méthode hybride de rendu située au cœur de nos travaux. Elle s'inscrit dans la lignée des méthodes de rendu de mésostructures combinant le principe de rasterisation et de lancer de rayon [47] et permet de découpler de manière souple et efficace le rendu de la macrostructure et de la mésostructure d'un objet. Bien que cette technique permette d'effectuer le rendu de mésostructure surfacique, elle se concentre principalement sur le rendu de détails volumiques de mésostructure contrairement à [47].
Dans cette section, nous proposons d'écaluer la méthode de rendu hybride que nous avons proposée, notamment en matière d'effets visuels et de performance (tels que décrits dans le chapitre 4, c'est-à-dire : la silhouette, la parallaxe, l'auto-ombrage et les interréfleions, les performances, la gamme de mésostructure et le coût mémoire.

La silhouette Notre méthode est capable de traiter correctement les silhouettes des mésostructures. En effet, cette méthode se base non pas sur une représentation planaire de la mésostructure mais procède effectivement une reconstruction totale des phénomènes liés à la visualisation, dans la même lignée que les modèles volumiques de mésostructure (section 4.1.2).

La parallaxe De la même manière que pour la silhouette, les effets de parallaxes sont gérés naturellement par notre méthode de *ray marching* local.

Auto-ombrage et inter-réflexions En ce qui concerne l'auto-ombrage et l'illumination (locale ou globale), il existe plusieurs méthodes qui dépendront des applications. Nous pouvons cependant considérer que le problème d'illumination locale et le problème de l'auto-ombrage peuvent être traités relativement efficacement par notre méthode. En premier lieu, notre méthode n'est pas basée sur une image mais sur une reconstruction de l'apparence lors du rendu. La problématique de l'illumination locale peut donc être reconstruite de manière classique lors du rendu en utilisant un modèle de Phong ou une BRDF arbitraire. En ce qui concerne l'auto-ombrage, ce problème est relativement voisin du problème de parallaxe. En effet, l'auto-ombrage consiste à déterminer si un point de la mésostructure est dans l'ombre, c'est à dire masqué par rapport à la lumière par un autre élément. Ceci relève de la même problématique que le phénomène d'auto-occlusion, qui consiste à déterminer si un point de la mésostructure est masqué par un autre par rapport à la direction de vue. Le principe de notre méthode peut donc être adapté à cette problématique en créant simplement un nouveau segment $[pe_{ombre}, ps_{ombre}]$ entre le point de la mésostructure considéré et la source de lumière. Notre méthode est donc adaptée pour traiter les phénomènes d'auto-ombrage. Les inter-réflexions et les effets de diffusion de la lumière sous la surface restent des problèmes vastes et plus complexes, qui dépassent le cadre de nos travaux. Ils ne sont donc pas traités ni abordés ici.

Performances Grâce aux évolutions du matériel graphique, le rendu de la macrostructure d'un modèle représente un coût négligeable en comparaison du rendu de la mésostructure en elle même. En effet, dans notre méthode ainsi que dans la plupart des méthodes basées sur un lancer de rayon présentées dans la section 4.1.1, la majorité des calculs nécessaires au rendu de l'apparence complexe sont effectués par fragment. Ceci présente l'avantage d'avoir une forte dépendance entre la taille occupée à l'écran par un objet et son coût de rendu (en terme de temps de calcul). Cette dépendance est la clé des performances des méthodes hybride basée sur une combinaison de rasterisation et de lancer de rayon, comme le *relief mapping*, et représente une souplesse par rapport aux méthodes basées uniquement sur la rasterisation. En effet, une modélisation explicite d'une mésostructure complexe, qui serait non seulement difficile à représenter mais aussi coûteuse en mémoire, aura nécéssairement un coût non négligeable lors du rendu. Par exemple, le nombre de polygones (dans le cas d'une représentation polygonale) pouvant être traité par la carte graphique, s'il est en constante évolution, reste néanmoins limité.

De plus, un objet occupera peu de pixels à l'écran s'il se trouve loin de l'utilisateur. Cela reviendrait donc à utiliser une partie de la puissance disponible de la carte graphique (en terme de nombre de primitives pouvant être traité par seconde) pour une importance visuelle faible. Alors que les méthodes à base de rasterisation doivent recourir à des méthodes complexes et coûteuses (en temps de calcul ou en stockage) pour simplifier l'objet afin de réduire le temps de rendu d'un objet recouvrant une zone de l'écran faible, les méthodes hybrides ont cette propriété naturellement. La figure 5.7 montre comment varient les performances de notre méthode en fonction du nombre de pixel recouvert par l'objet. Ces mesures ont été prises en effectuant le rendu d'un modèle composé de 8 000 points. Afin de rester général, la fonction évaluant la contribution de chaque échantillon (la fonction évaluation() de l'algorithme 5.2) à été réduite à un minimum : elle consiste à assigner une valeur constante de couleur et d'opacité à l'échantillon. Ces courbes soulignent la correlation entre les performances et la taille de l'objet à l'écran (avec un taux d'échantillonnage fixe) pour le cas surfacique (en haut) et volumique (en bas). Chaque courbe représente l'évolution de des performances en fonction de la taille de l'objet à l'écran en fixant un taux d'échantillonnage particulier pour chaque segment $[pe_i, ps_i]$.

Comme mentionné plus haut, le nombre effectif d'échantillons ainsi que le coût de calcul de chaque évaluation ont un impact direct sur les performances de notre méthode. La figure 5.8 montre l'évolution des performances de notre méthode en fonction du nombre d'échantillon, pour un nombre de pixel recouvert constant. Ces mesures ont également été prises avec un modèle de 8000 points et une fonction d'évaluation constante.

Comme on peut le constater, les deux facteurs ayant une influence sur notre méthode sont, indépendamment de la fonction utilisée pour évaluer la mésostructure, le nombre de pixels recouvert par l'objet et le taux d'échantillonnage d'un segment. Ce dernier paramètre nous donne en outre la possibilité de bénéficier d'un mécanisme de niveau de détails souple et efficace propre à la mésostructure. En effet, le taux d'échantillonnage peut être facilement adapté afin de réduire ou d'augmenter le nombres d'échantillons générés pour un segment $[pe, ps]$, permettant ainsi de faire varier les performances et la qualité visuelle de notre méthode de rendu.

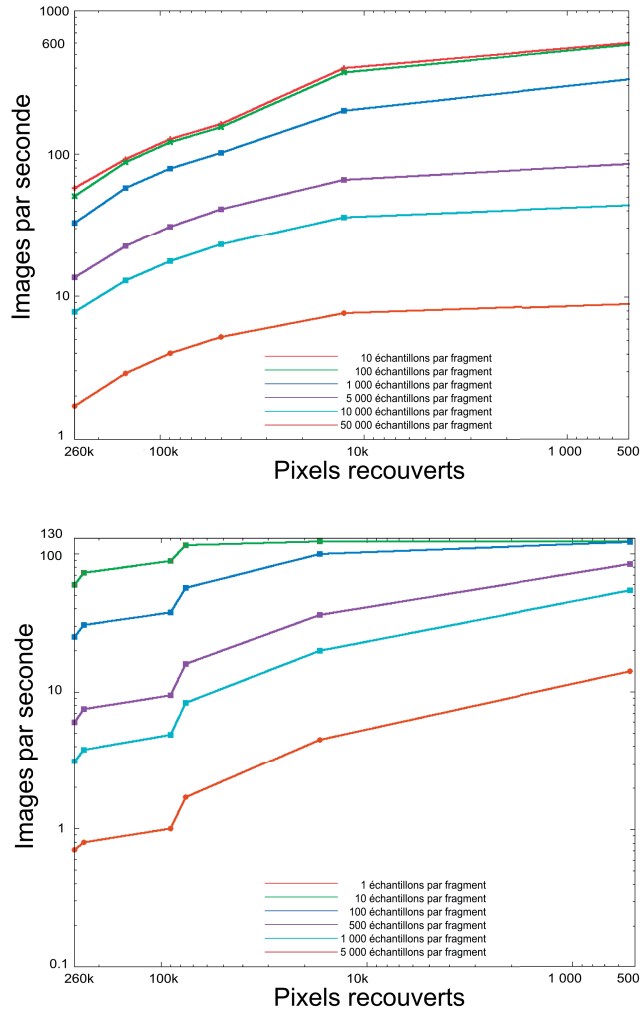


FIGURE 5.7 – Ces premières courbes décrivent pour un objet défini de manière surfacique (en haut) ou volumique (en bas) les performances de notre méthode pour le rendu d'un mlodèle de cube en fonction du nombre de pixels recouverts à l'écran. La fonction de chaque échantillon au résultat finale est l'assignation d'une constante.

Coût mémoire Notre méthode se basant sur une reconstruction de l'apparence de la mésostructure lors du rendu et non sur un précalcul ou un ensemble d'images, son coût mémoire correspond au coût de la représentation de la mésostructure en mémoire.

Gamme de mésostructure Cette méthode est capable de traiter une définition volumique de la mésostructure et reste donc suffisamment générale pour s'adresser à une vaste gamme de mésostructure. Il est possible de repré-

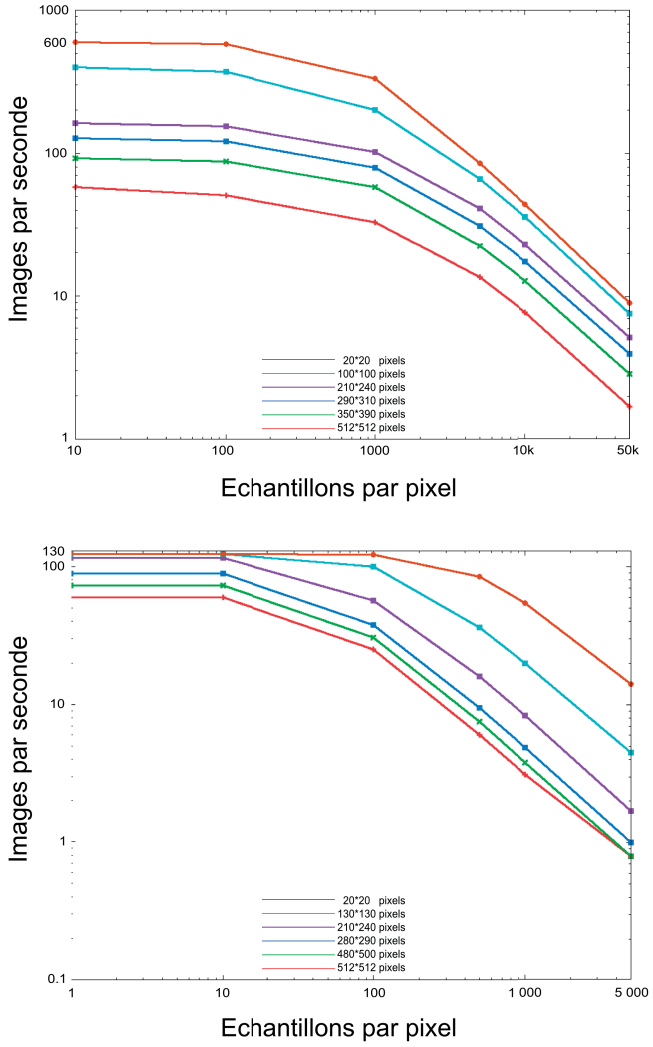


FIGURE 5.8 – Ces courbes décrivent pour un objet défini de manière surfacique (à gauche) ou volumique (à droite) l'impact sur les performances du nombre d'échantillons pour un rayon de vue. La contribution de chaque échantillon au résultat finale est ici définie par une constante.

senter de nombreux effets (naturels ou non) difficilement représentables en utilisant d'autre méthodes. Le chapitre 7 propose une étude plus détaillée des phénomènes naturels qu'il est possible de représenter grâce à cette méthode, et particulièrement en utilisant une définition procédurale.

Conclusion

Nous avons présenté dans ce chapitre une approche de rendu hybride, visant principalement une représentation surfacique et une représentation volumique. La méthode de traitement volumique présente des performances moindres comparée à la méthode surfacique. Ceci est principalement dû au sur-échantillonnage inhérent au recouvrement des noyaux. Il est à noter toutefois que la méthode surfacique, même avec un sur-échantillonnage n'est jamais équivalente à la méthode volumique. En effet, lors du rendu d'objets volumiques, les contributions des segments $[pe_i^G, ps_i^G]$ de chaque noyau G sont pondérées et mélangées. Ceci implique que la contribution de chaque point du volume de l'objet est composée de la contribution de ce point précis ainsi que la contribution (pondérée) de tous ses voisins. Ce phénomène est équivalent à l'utilisation d'un filtre gaussien en lancer de rayon volumique traditionnel et génère un anti-aliassage naturel. Un des autres effets de ce filtre est de nous permettre de nous abstraire du problème des discontinuités survenant au raccord entre deux primitives. Ce mélange permet également de traiter des mésostructures non périodiques, ou ne se raccordant pas parfaitement entre deux primitives voisines. Cependant, la méthode volumique souffre de deux inconvénients : des performances moindres et un problème de visibilité inhérent aux méthodes de splatting (voir section 3.3.2).

Une des perspectives majeures pour l'amélioration des performances de notre méthode consiste en l'utilisation d'un échantillonnage adaptatif pour la partie de *ray marching* local, dans le but de réduire le nombre d'échantillons nécessaires pour afficher une mésostructure. En effet, à l'instar des méthodes de *ray marching* traditionnelle, un schéma d'échantillonnage adapté en fonction des données de mésostructure peut conduire à des performances accrues. Il est également possible de tirer parti de la souplesse de notre méthode et de définir un schéma d'échantillonnage différent pour des rayons voisins, en se basant par exemple sur la position sur l'écran du rayon, sur l'éloignement de l'objet auquel il est rattaché ou sur la taux de rafraichissement souhaité pour l'application.
Notre méthode hybride présente toutefois les mêmes limitations que les méthodes basées sur le principe rasterisation/lancer de rayons. En effet, les fragments associés aux rayons de vue définissant la mésostructure sont générés par la rasterisation du modèle et ne peuvent être déplacés. Cela implique que les détails de mésostructure ne peuvent pas sortir de l'enveloppe externe de l'objet et conduit à limiter les phénomènes représentés. Nous considérons toutefois que de tels détails ayant une taille importante par rapport à la forme globale de l'objet relèvent plutôt de la macrostructure.

Chapitre 6

Un modèle deux niveaux pour le rendu par *splatting* de grands volumes de données

Dans ce chapitre, nous abordons l'ajout de détails de mésostructure volumique définis de manière « explicite » pour une application de rendu volumique. L'idée n'est pas d'ajouter une mésostructure quelconque à un modèle volumique, mais d'utiliser la méthode hybride décrite au chapitre 5 pour en augmenter les performances. En effet, de nombreuses méthodes parviennent à effectuer le rendu de modèles surfaciques complexes avec des performances accrues en séparant la mésostructure de la macrostructure (voir section 4.1.2). L'enjeu ici est d'utiliser le même découplage afin d'accroître les performances du rendu d'un objet volumique complexe modélisé par un très grand nombre de points (ou *splats*, voir section 2.1.3) par rapport à une méthode de rastérisation de type *splatting*. Cela consiste à créer pour un modèle volumique un nuage de points simplifié et un ensemble de textures 3D (la mésostructure) qui seront appliquées lors du rendu par notre méthode hybride.
Nous commençons par rappeler le contexte du rendu volumique par points (section 6.1), avant de présenter par la suite le découplage macro/mésostructure d'un modèle volumique (section 6.3). Le concept de base de ce découplage est d'établir un critère de décision permettant de classifier un ensemble de primitives comme « détails de mésostructure ». Nous finirons ce chapitre en présentant les résultats de cette applications ainsi que quelques pistes pour améliorer le découplage effectué.

6.1 Contexte de l'application

Le cadre de cette application est la visualisation de jeux de données volumiques de grande taille. Comme présenté dans la section 3.3.2, plusieurs méthodes de rendu existent pour permettre la visualisation interactive de données volumiques. La méthode du *splatting*, par exemple, est naturellement très

adaptée pour traiter des jeux de données non structurés de très grande taille avec beaucoup d'espaces vides. Ceci est principalement dû à l'efficacité de la représentation par points pour définir des données non-structurées (voir section 3.3.2). Toutefois, la méthode de *splatting* trouve ses limites lorsque le nombre de splats est très important. En effet, le nombre de primitives à traiter peut devenir un facteur limitant pour les méthodes d'ordre objet, qui ont pour principe de projeter les éléments d'un modèle sur l'écran. De plus, cette méthode peut effectuer des calculs inutiles si aucune optimisation n'est mise en place. Par exemple, les éléments qui sont masqués doivent néammoins être traités, ce qui conduit parfois à limiter encore plus les performances.

Une des méthodes classiques pour garder des performances interactives consiste à transformer le jeu de données non structuré en un jeu de données structuré (une grille régulière par exemple) en le rééchantillonant à une plus faible résolution. Ceci permet de stocker les données dans une texture 3D et d'en effectuer le rendu grâce à une technique adéquate (à base de *slicing* ou de lancer de rayon). Toutefois, cela introduit quelques problèmes inhérents à la nouvelle représentation des données. Par exemple, le problème de la mémoire disponible dans la carte graphique intervient à nouveau. En effet, stocker des données non-structurées dans une grille régulière (une texture 3D) implique souvent un coût mémoire plus important (on stocke dans la texture les espaces vides). L'utilisation d'une structure optimisée, comme un *octree* par exemple, permet de remédier à ce problème de stockage. Cependant, l'implémentation efficace d'une telle structure sur le GPU n'est pas aisée et cause une réduction significative des performances. En effet, ces structures nécessitent un nombre important d'indirections, réalisés sur le matériel graphique par des accès à une texture. Ces accès répétés sont la cause de ces baisses de performances, qui contrebalancent souvent les avantages en termes de stockage de ces structures. Par ailleurspour des très gros volumes, la taille mémoire peut rester trop importante, ce qui conduit nécessairement à dégrader plus ou moins fortement les données.

Une autre solution permettant de remédier à ce problème de complexité, tout en gardant les avantages d'une représentation non structurée, consiste par exemple à proposer une représentation multi-échelle des données originelles. Nous présentons quelques méthodes permettant de traiter ce problème dans la section suivante (section 6.2). Malgré une représentation multi-échelles, les performances peuvent être faibles. L'idée dans ce cas est d'utiliser la représentation multi-échelle pour « dégrader » la qualité de la visualisation, au prix de la perte de détails visuels à petite échelle. Cela revient à arrêter la représentation à un niveau plus élevé dans l'arbre hiérarchique.

Nous proposons dans ce chapitre d'assimiler ces détails, qui pourait être enlevés par une méthode de simplification, à des détails de mésostructure. De la même manière, nous proposons de considérer le jeu de données volumiques résultant d'une simplification comme une macrostructure. Nous nous retrouvons de ce fait dans une problématique identique à celle décrite dans le chapitre 5, à savoir le rendu de mésostructure volumique. L'objectif est donc d'utiliser notre méthode d'ajout de détails de mésostructure pour visualiser un jeu de données volumiques de grande taille avec des performances interactives et supérieures à un rendu par méthode de *splatting* du jeu de données brut, tout en préservant une qualité visuelle élevée.

Pour ce faire, nous proposons apres une présentation rapide des méthodes existantes de représentation hierarchique de jeu de données volumiques (section 6.2) une discussion sur la notion de détails en représentation volumique ainsi que la méthode que nous avons retenu pour séparer les détails de mésostructure de la macrostructure d'un objet volumique dans la section 6.3.

6.2 Représentation hiérarchique et simplifications

Nous avons présenté au cours de la section (section 3.1.2) la problématique posée par la visualisation de modèles définis de manière volumique. Nous avons également pu voir le principe sous-jacent aux méthodes de rendu volumique. De la même manière nous avons esquissé une présentation des méthodes de *splatting*, connues pour la qualité des images générées et pour leur traitement efficace des jeux de données non-structurés de grande taille. Nous allons aborder dans cette section quelques méthodes visant à augmenter les performances de ces méthodes, particulièrement en procédant à une réduction du nombre de splats projetés.

Mueller et al. [109] proposent par exemple un schéma basé sur l'opacité pour éliminer rapidement les splats masqués par d'autres éléments du volume. Dans [84], Laur et al. proposent une méthode hiérarchique très proche de notre méthode. L'idée est d'approximer une région d'un volume en analysant la variation de ses données. Ils proposent notamment d'approximer une région présentant une faible variation par un seul splat recouvrant cette région. Cela revient à faire disparître les détails de petite taille, améliorant ainsi les performances au détriment de la qualité visuelle. Alors que cette méthode est efficace pour certains jeux de données, des volumes avec des variations de données à

hautes fréquences sont peut adaptés à une telle simplification, en raison de la perte de données introduite.

Pour résoudre le problème de performances des méthodes de *splatting* lors du rendu de modèles comportant un grand nombre de splats et le problème d'efficacité et de stockage des méthodes de lancer de rayon lors du rendu de jeux de données non-structurées, les méthodes hybrides apparaissent comme une solution naturelle. Ma et al. [100] proposent une méthode reposant sur un choix entre une méthode de *splatting* ou basée sur une texture pour chaque région d'un modèle volumique. Toutefois, leurs travaux se focalisent sur la visualisation de très larges jeux de données issus de simulation physique et sont peu adaptés au rendu haute qualité de modèle classiques. De manière équivalente, Wilson et al. [168] introduisent un schéma hybride basé sur la projection de splats semi-opaques en conjonction avec une technique basée sur une texture 3D représentant une version sous-échantillonnée du modèle. Alors que cette méthode permet le rendu de jeux de données large et non structurés, elle procède à une simple addition des contributions , ce qui peut résulter en des erreurs d'approximation et nécessiter une intervention manuelle afin d'éviter les artefacts visuels. Plus récemment, Kaelher [129] propose une représentation hybride pour le stockage et le rendu de modèles volumiques, en simplifiant lors du rendu les régions larges et distantes du jeu de données par une simple primitive point.

D'une manière générale, ces méthodes proposent lors du rendu un choix entre une méthode basée soit sur des splats soit sur une texture 3D pour chaque région. Ce que nous proposons dans ce chapitre est une méthode permettant d'utiliser effectivement les deux méthodes pour une même région, chaque splat contenant une texture 3D (la mésostructure).

6.3 Notion de détails pour une représentation volumique

L'idée de cette application est de réduire la complexité en terme de primitives d'un objet volumique sans réduire sa complexité visuelle. Pour cela, nous proposons d'utiliser notre méthode d'ajout de détails de mésostructure volumique (voire section 5.1.2). Cela implique de diviser l'objet volumique en question en deux niveaux, l'une représentant la macrostructure, l'autre représentant les détails de mésostructure.

La question qui se pose alors est la suivante : étant donné un sous-ensemble d'un objet volumique, est-on capable de déterminer si ce sous-ensemble appartient à la mésostructure ou à la macro-structure de l'objet ? Alors que nous avons évoqué dans l'introduction quelques notions pour différencier la macrostructure de la mésostructure, la frontière entre les deux peut être difficile à déterminer automatiquement, en particulier dans le cas d'objets définis de manière volumique. Le point central de cette application consiste à définir un critère de décision permettant de répondre de manière efficace à la question posée précédemment. Nous proposons dans la section suivante une première approche concernant un critère de décision.

Munis de ce critère, nous pouvons diviser un modèle volumique en utilisant un schéma hiérarchique. Le principe est d'évaluer un sous-ensemble du modèle avec notre critère de décision. Cette évaluation indique si ce sous-ensemble peut être considéré comme du détail de mésostructure ou non. Si cette partie du modèle n'est pas considérée comme du détail de mésostructure, nous choisissons alors de la subdiviser et répétons cette évaluation sur chacune des sous-parties nouvellement créées. Dans le cas où un sous-ensemble est évalué positivement avec notre critère (c'est-à-dire est un détail de mésostructure), deux étapes prennent alors place :

– Premièrement, la sous-partie en question doit être stockée quelque part. En effet, nous ne voulons pas perdre les informations relatives à la richesse visuelle du modèle. Les données en question étant issues d'un modèle volumique défini de manière explicite, nous choisissons de les stocker sous la forme d'une texture 3D. C'est-à-dire que nous créons pour chaque sous-partie une texture 3D contenant les informations propres à cette sous-partie. Ces informations sont stockées sur le matériel graphique et utilisées au cours du rendu. La texture 3D nouvellement créée doit avoir une résolution au moins identique à la résolution de la sous-partie d'origine, afin de ne pas perdre de détails. Toutefois, il peut être envisageable, dans le cas de détails de mésostructure uniforme, de réechantilloner cette texture 3D afin de gagner de l'espace mémoire sans perte de détails.

– Dans un second temps, nous définissons une représentation plus grossière pour cette sous-partie. En effet, le but de cette phase de décomposition est d'obtenir une représentation de la macrostructure du modèle. Pour cela, il est nécessaire de définir une approximation des informations de mésostructure d'origine, afin de rester visuellement proche du modèle. Nous choisissons de supprimer du modèle les informations « ex-

plicites » modélisée par les splats (les informations ont été stockées dans une texture 3D au cours de l'étape précédente) et de les remplacer par un unique échantillon qui aura comme information de densité la moyenne des densités des détails de mésostructure. De plus, nous associons à cet échantillon un noyau de reconstruction (à l'instar de l'*EWA splatting*, une gaussienne 3D radialement symmétrique tronquée) suffisamment large pour couvrir l'ensemble de la sous-partie considérée du modèle. De manière à pouvoir reconstruire au cours du rendu les informations de mésostructure, cet échantillon dispose également d'un pointeur vers la texture 3D contenant les informations de détails (stockées sur la carte graphique au cours de la première étape).

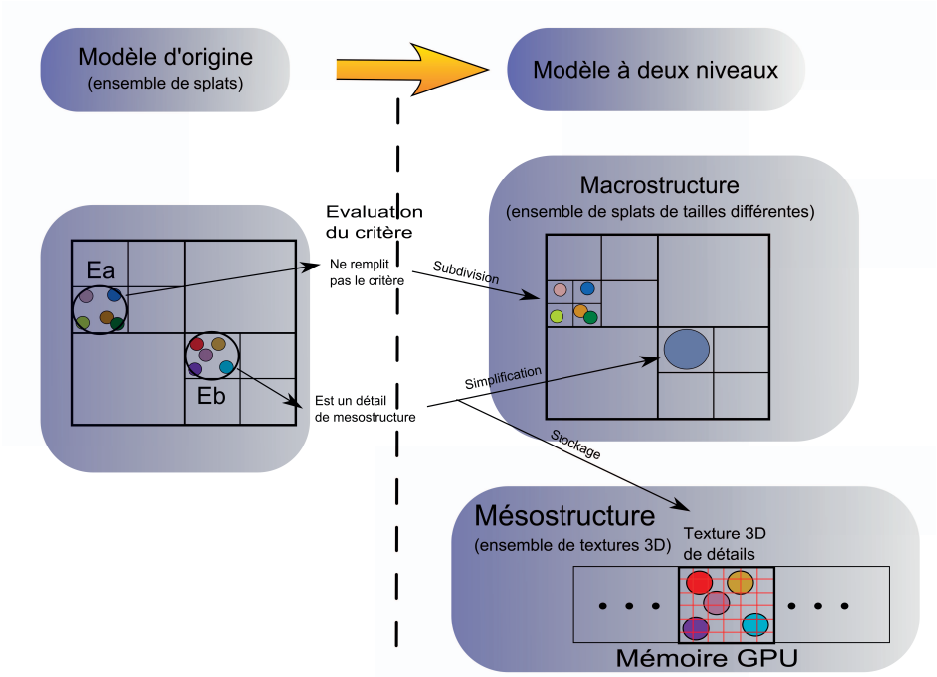


FIGURE 6.1 – Décomposition d'un jeu de données volumiques en un modèle à deux niveaux, composé d'un ensemble de splats (la macrostructure) chacun associé à une texture 3D (la mésostructure).

La figure 6.1 illustre (en 2D) le processus de décomposition en un modèle à deux niveaux. Chaque cellule de la structure hiérarchique est évaluée. Si l'évaluation est négative (Ensemble *Ea* sur la figure), la cellule est subdivisée et chaque cellule fille est réévaluée. Si le critère est positif (ensemble *Eb*), la cellule est approximée par un nouvel échantillon. Les informations de la cellule

sont quant à elles stockées dans une texture 3D sur la carte graphique.
Pour des raisons de simplicité, nous avons choisi d'effectuer cette décomposition en nous basant sur un modèle hiérarchique d'*octree*[1]. Dans notre modèles à deux niveaux, nous remplaçons un ensemble de splats de même dimension par un ensemble de splats de taille variable chacun muni d'une texture 3D.

6.3.1 Critères de décision

Nous abordons ici les critères de décision pour la décomposition d'un modèle volumique. Tout d'abord, nous avons comme données d'entrée un ensemble d'échantillons dans l'espace, auquels sont associés une valeur scalaire de densité. Nous avons également à notre disposition une fonction de transfert permettant d'associer cette valeur à une couleur et une opacité (section 3.1.2).

Le critère principal que nous avons défini se base sur l'apparence visuelle de la sous-partie du modèle à diviser. C'est un critère de *variation visuelle*. Il se base sur le constat suivant : notre méthode utilise une technique de lancer de rayons pour la mésostructure et une technique de *splatting* pour la macrostructure. Ces deux schémas (du moins leur implémentation sur le matériel graphique) reposent sur des principes différents et donnent pour certaines régions des résultats légèrement différents. En effet, la technique du splatting (GPU) peut souffrir d'un problème de visibilité. Les échantillons sont de manière générale regroupés en tranches, chaque tranche étant traitée de manière séquentielle et les échantillons d'une même tranche étant mélangés (voir section 3.3.2). Ceci peut conduire à des problèmes de visibilité lorsque le nombre de tranches n'est pas suffisant. C'est souvent le cas lors d'applications interactives. Toutefois, cette méthode offre un rendu visuel de qualité grâce à son filtre de reconstruction efficace. Le lancer de rayon GPU, de manière inverse, n'a pas de problème de visibilité mais utilise le filtre d'interpolation trilinéaire de la carte graphique, ce qui peut conduire à des effets d'escalier lors du rendu[2].
Les différents avantages et inconvénients de ces deux méthodes sont à prendre en compte ici. Malgré toutes les optimisations existantes, une méthode de lancer de rayon sur GPU basée sur une texture 3D ne peut traiter le problème des grand espaces vides dans un jeu de données aussi rapidement qu'une méthode de *splatting*, qui possède cette propriété naturellement de par sa représentation. Ceci est également valable pour des régions ayant une faible variation,

1. Le modèle d'*octree* consiste, en 3 dimensions, à diviser une cellule en 8 parties égales selon les axes.

2. Ces effets peuvent être atténués en utilisant un suréchantillonnage ou un schéma d'interpolation plus sophistiqué mais cela implique souvent une baisse de performance.

qui peuvent être rendues rapidement par *splatting* (l'approximation du tri devient moins problématique lorsque les données varient faiblement). Inversement, une région contenant beaucoup de détails contigüs sera rendue plus rapidement par lancer de rayon sur GPU, en raison du problème (majeur dans ce cas) de visibilité, traité naturellement par de telles méthodes. L'idée est de choisir la méthode la plus adaptée en terme de rapidité pour chaque région.

Notre critère de décision revient à décider si une région (un sous-sensemble compact) d'un modèle sera rendu avec une technique de *splatting* (il appartient à la macrostructure) ou une technique de lancer de rayon (il appartient à la mésostructure). Toutefois, les différences de rendu entre les deux techniques peuvent provoquer des artefacts visuels. Ces artefacts peuvent de plus s'accumuler et perturber la qualité du résultat final. L'idée de ce critère de variation visuelle est de minimiser ces artefacts. Pour ce faire, nous prendrons comme hypothèse qu'un rendu à base de *splatting* (logiciel), de par la qualité de son filtre de reconstruction, produit des résultats visuellements corrects et de meilleure qualité. Partant de cette hypothèse, notre critère de variation visuelle doit donc chercher à minimiser les différences entre un rendu à base de *splatting* et un rendu à base de lancer de rayon. D'où l'idée de ce premier critère : *On considère une région du modèle comme un détail de mésostructure s il est possible d effectuer le rendu de cette région avec une technique de lancer de rayons GPU sans introduire de grandes différences visuelles par rapport à un rendu à base de splatting et si ce rendu est plus rapide, ce qui est toujours le cas au delà d un certain nombre de splats.*

De manière à détecter ces régions, il est nécessaire d'introduire une métrique qualitative d'erreur mesurant la variation entre les deux schémas de rendu. Cette variation v_r est exprimée pour un rayon de vue r traversant le volume. Nous avons choisi de définir cette variation comme *la distance dans l espace couleur*[3] *entre les deux couleurs obtenues pour l évaluation de l équation du rendu volumique 3.3 le long du rayon de vue r en utilisant les deux techniques de rendu :*

$$v_r = |C_{splat} - C_{raycast}| \tag{6.1}$$

La couleur $C_{raycast}$ est obtenue par l'évaluation de l'approximation discrète de l'intégrale du rendu volumique en utilisant une technique de lancer de rayon basée sur un schéma d'interpolation trilinéaire. Nous effectuons pour l'instant cette évaluation avec un nombre arbitraire d'échantillons le long du rayon de vue. La couleur C_{splat} est quant à elle calculée par une méthode de *splatting*.

3. Ce critère peut être mesuré dans n'importe quel espace couleur ($RGBA$ ou $L*a*b$ pour une variation linéaire avec la perception).

Cette valeur v_r reflète les différences visuelles entre une solution de rendu à base de *splatting* ou de lancer de rayon (avec un schéma d'interpolation linéaire) pour un rayon de vue r arbitraire. Ceci peut être considéré comme une estimation des artefacts visuels que l'on obtiendrait si on passait d'un schéma à l'autre. Dans un souci de simplicité, nous avons choisi de considérer cette valeur telle quelle. Il est bien sûr à noter que cette simple valeur ne reflète pas l'ensemble des artefacts visuels pouvant être détectés par un l'œil humain le long d'un rayon de vue. Il conviendrait pour obtenir un critère plus efficace de se rapprocher des nombreux travaux étudiant la perception de l'œil humain. Par exemple, l'œil étant plus sensible aux contours, une différence de couleur sur un rayon de vue affleurant une propriété visuelle caractéristique aura un impact plus important qu'un autre. De telles études sortent du cadre de nos travaux mais offrent une perspective non négligeable pour améliorer la qualité des résultats. Nous allons cependant considérer cette variation de couleur comme une approximation des artefacts visuels du résultat final.

Nous avons choisi d'exprimer la variation visuelle V dans une région contigüe comme la valeur moyenne du carré des variations le long de tous les rayons de vue traversant la région :

$$V = \int_{\Omega} v_r^2 \partial\Omega \tag{6.2}$$

avec Ω la sphère de toutes les directions de vue possibles.
Cette intégrale ne peut être directement calculée pour notre modèle explicite. Nous choisissons de l'approximer avec une somme discrète :

$$V \approx \sum_{r=0}^{N} v_r^2 \tag{6.3}$$

avec N une discrétisation arbitraire de la sphère Ω.
Alors que cette métrique permettrait d'obtenir une mesure de la variation visuelle sur une région contigüe, il est difficile (et coûteux en temps de calcul) d'évaluer cette équation pour un grand nombre N de rayons de vue. Pour obtenir néanmoins une métrique, nous utilisons une distribution stochastique et évaluant l'équation (6.3) avec seulement un sous-ensemble de rayons de vue. La valeur de ce paramètre N a toutefois un impact critique sur notre méthode de simplification. Intuitivement, la valeur de N est directement corrélée à l'exactitude de la mesure ainsi qu'au temps de calcul nécessaire pour évaluer l'équation 6.3. Au stade de ces travaux, N est choisi de manière empirique au travers d'expérimentations. Il est bien sûr tout à fait envisageable de définir une méthode déterminant de manière automatique une valeur de N. Il est par exemple possible de définir une valeur de N en analysant les données de la région à évaluer. Intuitivement, dans une région présentant des variations de

densité à haute fréquence, deux rayons aux directions de vue proches auront des résultats néanmoins très différents. Une mesure efficace d'une telle région ne pourra être obtenue avec une valeur de N trop faible car des détails (potentiellements visibles) risqueraient de ne pas être pris en compte.
Cette évaluation de la différence visuelle est complexe et repose sur un précalcul. Toutefois, cela pose le problème pour certaines applications de l'édition de la fonction de transfert. En effet, des applications, principalement dans le domaine médical, nécessitent une édition interactive de la fonction de transfert, afin de masquer (par la transparence) ou de rendre visible diverses parties d'un modèle. Il n'est pas possible avec ce critère de modifier la fonction de transfert tout en gardant une application interactive, même avec des valeurs de N faibles, car le calcul de la décomposition du modèle à deux niveaux prendrait trop de temps. Pour être exact, bien que la génération de la structure d'*octree* soit rapide, les calculs nécessaires à l'évaluation de notre critère peuvent difficilement être réalisés en temps interactifs.

Afin de permettre une édition interactive de la fonction de transfert, un critère de décision pouvant être évalué rapidement doit être mis en place. Une première approche de ce nouveau critère empirique nous conduit à analyser la variance d'une zone. La variance étant une mesure de la dispersion des densités par rapport à une valeur moyenne, une zone à variance faible indique que les densités ont des valeurs très proches les unes des autres dans cette région. Il est donc possible d'approximer cette zone par un seul splat et d'effectuer le rendu précis avec une méthode de *ray marching*. Inversement, une zone avec une forte variance peut difficilement être approximée par un seul splat et doit donc être subdivisée.

En pratique, ces deux critères ont pour effet de « grouper » les échantillons des zones au contenu similaire, créant ainsi une représentation effectivement grossière d'un modèle volumique. Toutefois, ces critères effectue la même hypothèse : ils négligent l'erreur de visibilité introduite par l'implémentation matérielle des algorithmes de *splatting* (voir section 3.3.2). En effet, afin de tirer parti des capacités de parallélisation des GPU, les échantillons du modèle sont regroupés en tranches parallèles au plan image. Tous les échantillons d'une tranche sont considérés comme étant à la même profondeur et leur contributions sont mélangées. Cela peut causer un problème de visibilité et un effet de flou lorsque les tranches ne sont pas assez nombreuses.

Dans le cas de nos applications, le nombre de tranches est une variable laissée à l'utilisateur final et n'est pas connue lors du pré-calcul. Selon notre critère analysant la variance, les zones présentant une haute variance ne sont

pas approximées et sont représentées par un ensemble de points identiques au jeu de données d'origine. Lorsque tous ces points se retrouvent à l'intérieur d'une même tranche, leurs contributions sont mélangées sans information de profondeur relative, comme dans le *splatting* traditionnel, ce qui crée parfois l'effet visuel de flou. Si cet effet est peu visible dans le cas de données ayant une variation de densité de faible fréquence, il peut créer de sévères artefacts visuels dans le cas de données variant à hautes fréquences. La solution naturelle consiste alors à augmenter le nombre de tranches de la méthode. Toutefois, cela contribue à réduire fortement les performances lors de la visualisation d'un jeu de données complexe.

Nous proposons d'améliorer le critère à base de variance pour prendre en compte ce phénomène. Nous avons jusque ici un jeu de données composé d'un ensemble de splats de taille variable chacun associé à une texture 3D. Ce qui caractérise une zone où l'information de visibilité est importante pour la qualité visuelle est principalement la variation fréquentielle de ses données. En effet, un contenu variant à haute fréquence a besoin d'échantillons triés finement afin d'être correctement représentée. De manière à augmenter les performances de notre application, il convient donc d'effectuer le rendu de ces zones par une méthode de *ray marching*, apte à traiter le problème de visibilité naturellement. Nous avons pour cela choisi empiriquement de définir une taille minimale pour les éléments de macrostructure, en fonction de la fréquence de variation des données. Les régions plus petites que cette taille minimale sont automatiquement converties en détails de mésostructure, stockées en mémoire graphique et approximées par un unique échantillon. Cela permet à l'utilisateur de choisir de manière empirique entre un rendu moins performant mais profitant de la qualité d'une méthode de *splatting* ou un rendu plus rapide mais potentiellement sujet à des artefacts de discrétisation dus à l'interpolation linéaire du GPU. Ce critère est défini empiriquement et est non optimal mais permet en pratique une réduction du nombre de splats pour la plupart des modèles.

6.4 Résultats et discussions

Nous présentons dans cette section quelques résultats portant sur la qualité visuelle et les performances de notre méthode hybride. Nous comparons principalement les résultats avec une méthode d'*EWA splatting* pour le même jeu de données volumiques. Les mesures ont été effectuées avec une GeForce 8800GTX et pour une résolution d'écran de 512×512 pixels. En premier lieu,

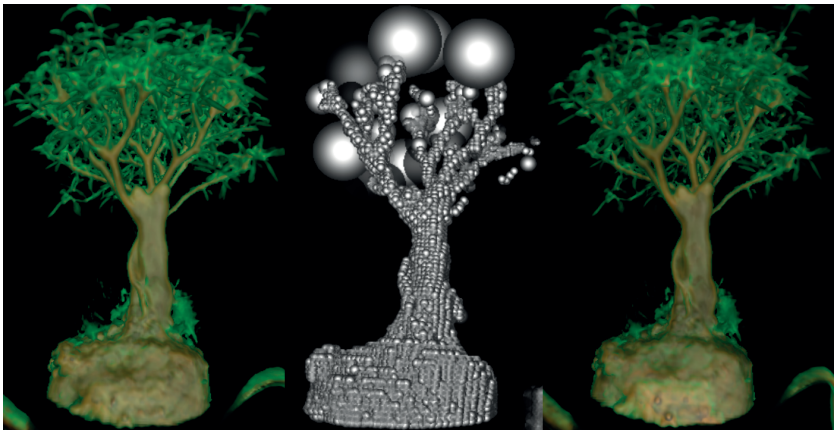


FIGURE 6.2 – Modèle d'arbre. À gauche : *EWA splatting*. 8M splats - 2.2 fps. Au milieu : notre représentation hybride. À droite : notre méthode hybride avec rendu de mésostructure explicite. 46k splats - 8.3 fps.

la figure 6.2 compare le rendu obtenu avec notre méthode (à gauche) avec un rendu à base d'*EWA splatting* (à droite) ainsi que les différentes performances. La figure du milieu illustre la macrostructure obtenue avec notre critère de variance. Pour plus de visibilité, les splats ayant une contribution nulle (une densité de 0) ont été enlevés de la figure (certaines zones du feuillage ont également été enlevées). Il est à noter que ces splats font néanmoins toujours parti du modèle et sont projetés sur l'écran lors du rendu. On peut observer que la macrostructure « colle » efficacement aux bords du tronc de l'arbre. En ce qui concerne le feuillage, c'est typiquement une région où la densité varie fortement d'un échantillon à un autre. Notre critère empirique permet de considérer cette zone comme un ensemble d'éléments de mésostructure et d'en effectuer le rendu par *ray marching*. Il est à noter toutefois que les paramètres empiriques de cet exemple ont nécessité beaucoup d'essais pour être définis.
Notre critère de variation visuelle réduit de manière efficace le nombre d'éléments de la plupart des modèles de 50 à 80%, sans perte visuelle importante. La figure 6.3 montre l'impact des différents seuils de qualité définis par l'utilisateur sur les performances et le résultat visuel de la méthode. Intuitivement, plus le seuil est élevé, plus les performances s'accroissent tandis que la qualité diminue (on accroît l'erreur d'approximation acceptée). Il est à noter ici qu'aucun schéma de pré-classification n'a été employé pour défausser les points du modèle ayant une densité nulle afin de rester plus général et de permettre le traitement de données volumiques où chaque point participe au résultat.

Sur la figure 6.4, nous montrons les différences principales entre les critères de variation visuelle et de variance. Alors que la plupart des propriétés

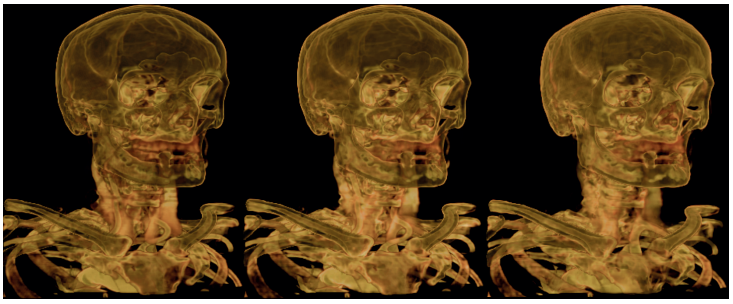


FIGURE 6.3 – Modèle volumique (512^3 voxels). À gauche : 2M points - $\epsilon_{\text{variation visuelle}}$ = 2% - 2.9 fps ; Au milieu : 386k points - $\epsilon_{\text{variation visuelle}}$ = 5% - 4.8 fps ; À droite : 250k points - $\epsilon_{\text{variation visuelle}}$ = 10% - 8.8 fps.

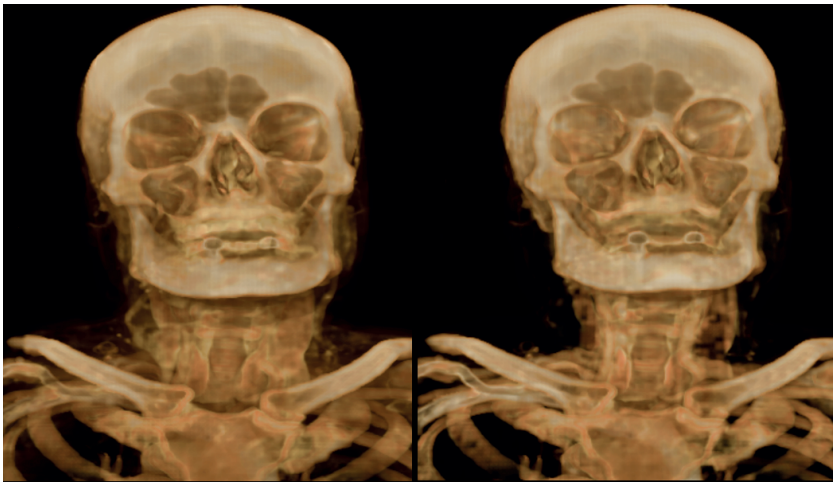


FIGURE 6.4 – Modèle rendu avec notre méthode (530k points) avec les deux critères proposés. À gauche : critère de variation visuelle. À droite : critère basé sur la variance.

visuelles caractéristiques sont préservées, les zones de faible variation de données, comme le cou du modèle, sont dégradées lors d'une représentation basée sur le critère de variance mais préservées lors de la représentation avec le critère de variation visuelle. Afin de juger de l'apport des détails de mésostructure sur un modèle, la figure 6.5 montre le même modèle, présenté avec et sans détails de mésostructure. Nous montrons également l'apport de notre méthode pour le rendu d'un jeu de données non structuré issu d'une simulation (voir figure 6.6).

Il est à noter que nous présentons dans cette section des jeux de données à base de voxels pouvant être rendus sans difficultés particulières par une méthode de type *ray marching* ou de *slicing* avec les capacités actuelles des GPU.

Mais le modèle le plus complexes (de taille 512^3) occupait une place extrêmement importante en mémoire graphique (dépassant la capacité de stockage), sur les modèles de cartes disponibles à l'époque où ces travaux ont été conduits (les premiers modèles de *GeForce 8*). On peut constater que notre méthode présente une augmentation importante des performances par rapport à une méthode de splatting, même pour des modèles « simples ». Cette différence s'accroît avec la taille des modèles.

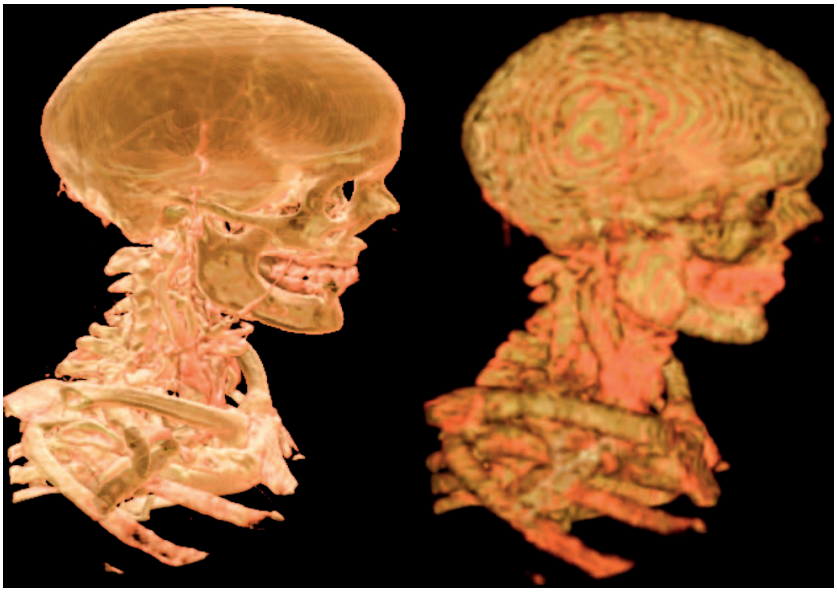


FIGURE 6.5 – Le modèle volumique rendu avec 2M de points avec ajout de détails de mésostructure (à gauche, 2.1 fps) et sans détails (à droite, 12.3 fps).

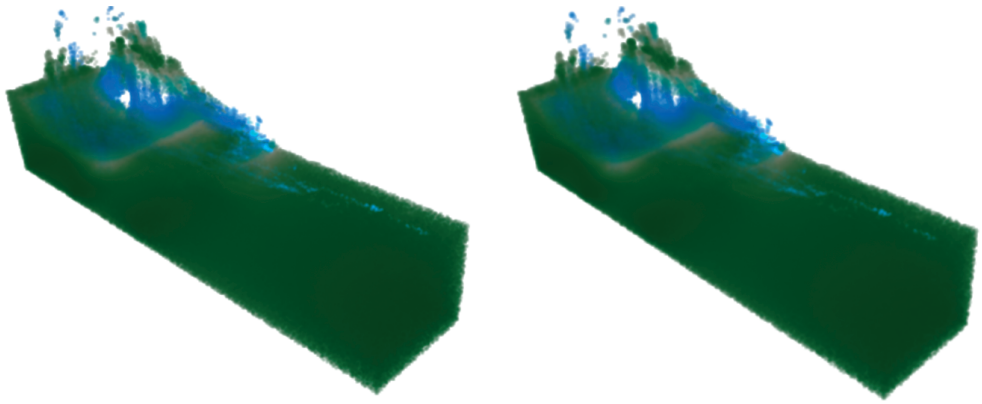


FIGURE 6.6 – Un exemple de données non structurées issues d'une simulation MAC. À gauche : rendu par *splatting* classique - 1.5M points - 11 fps. À droite : avec notre méthode hybride - 250k points - 16 fps.

6.5 Conclusion

Nous avons présenté dans ce chapitre la première application de notre méthode de rendu hybride décrite au chapitre 5. Cette application repose sur un découpage d'un modèle volumique de grande taille en informations de macrostructure représentées par un nuage de points et un ensemble de textures 3D représentant les détails explicites de mésostructure. Cela permet en pratique d'augmenter les performances des méthodes à base de *splatting*. Les performances de notre application, comme pour le *splatting* « classique » restent toutefois inférieures aux techniques de rendu à base de texture 3D. Cependant, notre technique devient rapidement intéressante dans le cas de très larges modèles, pour lesquels le stockage en mémoire GPU sous forme de texture 3D devient problématique.

Il est toutefois possible d'améliorer cette méthode. Afin d'accroître son efficacité, la première piste serait l'utilisation d'un schéma hiérarchique de simplification plus efficace qu'un *octree*. En effet, l'*octree* à pour particularité de diviser l'espace de manière régulière, ce qui ne correspond pas particulièrement à tous les jeux de données (particulièrement les jeux de données présentant des formes sphériques). Nous pensons que l'utilisation d'une technique permettant de mieux approximer les données, comme une technique à base de RBF (de l'anglais *radial basis functions*) par exemple, serait plus efficace et permettrait d'accroître à la fois les performances et la qualité des images. Une des idées principale est également d'utiliser un échantillonnage adaptatif pour la méthode de *ray marching*. En effet, un tel schéma adaptatif du taux d'échantillonnage, jusqu'ici constant pour toute l'image, permettrait de diminuer ou d'aug-

menter le nombre d'échantillons générés pour chaque fragment. Ceci permettrait d'augmenter également la qualité et les performances de la méthode. Un tel schéma peut être défini en tenant compte de diverses variables, comme la taille ou la visibilité d'une région (une mesure de l'importance visuelle de la région sur la scène).

En ce qui concerne nos critères de variation, il reste encore de nombreux points à étudier, comme leur définition automatique par exemple. En effet, la définition de la valeur des différents seuils d'erreur, effectuée par méthode d'essai-erreur, reste difficile et une plus grande automatisation permettrait d'accroître la pertinence de notre méthode. Une des pistes principales réside dans une décomposition en ondelettes du jeu de données. L'efficacité à analyser les discontinuités et phénomènes locaux dans les signaux de cette décomposition devrait permettre une décomposition plus efficace de notre modèle. Le dernier point d'amélioration à citer est une analyse complète de l'erreur introduite par ce modèle. En effet, les approximations introduites par notre découplage sont bornées pour une région (avec le critère de variation visuelle) mais ces erreurs d'approximations peuvent s'accumuler sur la même zone de l'écran, créant ainsi une erreur qui peut potentiellement devenir plus élevée. Une analyse de notre métrique pourrait dans ce cas conduire à une meilleure définition des paramètres et fournir une approximation de la variation visuelle totale dans l'image finale.

Finalement, notre méthode de rendu de détails volumiques explicites peut être dérivée pour une utilisation permettant de texturer de manière arbitraire un objet. En effet, l'utilisation de splats associés à une texture 3D pourrait permettre l'application de motifs répétitifs stockés dans une unique texture afin d'enrichir l'apparence visuelle d'un modèle volumique. Une telle application, proche du concept des *lapped textures*[127] de par l'utilisation du filtre de reconstruction permettant le « mélange » des contributions de chaque splat, permettrait d'effectuer le rendu de structures complexes, comme un feuillage, une pelote de laine, ou encore des barbelés.

Chapitre 7

Modélisation et rendu de mésostructures procédurales par déformation d'espace

Nous présentons dans ce chapitre une méthode d'ajout de détails de mésostructure volumique définis de manière procédurale. Les modèles définis de manière procédurale offrent plusieurs avantages par rapport à des modèles définis explicitement. Premièrement, une définition procédurale n'est pas liée aux problèmes de discrétisation propres aux définitions explicites. Ceci augmente grandement la précision de tels modèles. Deuxièmement, il est également possible grâce à une telle définition de représenter des modèles ayant une taille infinie sans les effets de répétitions communs aux modèles explicites. Deuxièmement, un des avantages indéniables d'une telle représentation est son extrême compacité. Contrairement aux modèles explicites, seul un petit jeu de paramètres est nécessaire pour reconstruire le modèle. Il existe toutefois quelques inconvénients par rapport à d'autres représentations. Naturellement, le principe d'une méthode procédurale étant de reconstruire un modèle à partir d'un jeu de paramètres, cela implique des calculs. Ces calculs peuvent varier en complexité mais il est assez évident que les représentations procédurales génèrent un surcoût en terme de calcul par rapport à des représentations explicites. Ceci implique pour beaucoup d'applications des performances généralement moindres. L'un des problèmes supplémentaire des définitions procédurales résulte en outre de sa définition même : alors qu'il est assez intuitif de modéliser un objet ou un phénomène de manière explicite, il est parfois difficile de corréler les règles et paramètres d'une représentation procédurale avec le phénomène ou l'objet à représenter.

Nous nous plaçons ici dans le contexte de détails de mésostructure reposant sur une représentation procédurale basée sur une fonction pseudo-aléatoire. Ceci présente l'avantage de pouvoir générer des détails au sein d'un volume infini sans effet de répétition. En effet, le caractère aléatoire d'une fonction de bruit conjointement avec un ensemble de paramètres génère des détails qui

« ressemblent » au phénomène que nous souhaitons modéliser sans pour autant être totalement identiques. Cette souplesse permet de donner une apparence arbitraire à un objet sans effets de répétition.
Pour être plus spécifique, cette application se base sur le principe des *hypertex-*

FIGURE 7.1 – Quelques exemples de phénomènes naturels complexes.

tures. Les hypertextures ont été définies il y a deux décennies par Perlin et Hoffert [123]. Elles représentent une technique d'ajout de détails volumétriques sur les surfaces. De façon simple, les hypertextures consistent en une déformation de l'espace 3D d'un objet. Cette déformation de l'espace, reposant sur une ou plusieurs fonctions de modulation de la densité (basées, dans notre cas, sur une fonction procédurale) induit une déformation de l'objet. Cette dernière peut complètement altérer la forme de l'objet ou résulter en une apparence de texture, comme de la fourrure ou de la mousse. Il est également possible de créer des gaz, de la fumée, des nuages ou des effets de flamme en ajoutant des détails de mésostructure à des modèles volumiques grossiers. Quelques exemples d'effets complexes du monde réel pouvant être potentiellement modélisés avec des hypertextures sont montrés sur la figure 7.1. Aucun de ces effets ne peut être facilement représenté en utilisant des techniques telles que le *bump mapping* ou le *displacement mapping*. En effet ces techniques se basent sur un déplacement des primitives composant la surface de l'objet en direction de la normale à la surface et sont donc incapables de représenter des gammes de déformations plus complexes, tels les déformations incluant des changements topologiques. La figure 7.2 présente trois classes de déformations.

Toutefois, il reste difficile de corréler la définition procédurale d'une hypertexture avec un résultat visuel donné. C'est à ce problème de contrôle et de prévision du résultat final que nous nous attaquons dans ce chapitre. Pour ce faire, nous proposons une reformulation du principe des hypertextures afin d'augmenter la corrélation entre le modèle procédural et le résultat final. Cette re-

FIGURE 7.2 – Trois classes de déformations : un simple *displacement mapping* (à gauche), une déformation de la surface par un champ de vecteurs arbitraire (au milieu) et une déformation volumique « complète » par un champ de vecteurs 3D avec des changements de topologie (à droite).

formulation se base sur l'utilisation d'une *fonction de transfert de forme*. Cette fonction de transfert permet à l'utilisateur de définir et de contrôler la forme finale de la déformation induite par le modèle d'hypertexture. De plus, notre méthode d'ajout de détails de mésostructure, compatible avec cette formulation, nous permet d'obtenir en temps réel un résultat visuel. Ceci nous permet de créer un outil de modélisation et de visualisation d'hypertextures intuitif et interactif.

Dans un premier temps, nous présentons le modèle général des hypertextures (sections 7.1 et 7.2). Nous entrons par la suite dans les contributions à proprement parler, c'est-à-dire notre nouvelle formulation du modèle d'hypertexture (sections 7.3). Avant de discuter des résultats de ce modèle, nous présentons au préalable (sections 7.4) quelques détails techniques quand à l'intégration de cette méthode dans notre schéma général d'ajout de détails de mésostructure (du chapitre 5).

7.1 Les hypertextures

Les hypertextures sont basées sur une forme générale de déformation ainsi que sur une fonction de modulation de la densité (*DMF*) définie dans un espace 3D. Le volume est déformé dans son ensemble, ce qui permet une modélisation de concavités et de trous et donc de changements de topologie. À l'instar des textures 3D, la définition de la DMF ne requiert aucune paramétrisation de la surface, ce qui donne une souplesse non négligeable à cette technique. De ce

fait, les problèmes d'auto-intersection ou de distorsion dus à la courbure de la surface ou à sa paramétrisation ne se posent pas. L'un des avantages naturels des hypertextures réside en sa définition procédurale, généralement basée sur un bruit 3D ou une turbulence. Ceci rend les méthodes à base d'hypertextures extrêmement compactes, à l'instar de toutes les méthodes procédurales. En outre, les hypertextures présentant une déformation basée sur un volume, les informations de semi-transparence des structures sont efficacement traitées, permettant ainsi le rendu d'effets complexes tels les nuages, le feu ou la fourrure.

Les hypertextures ont souvent été appliquées en conjonction avec des surfaces implicites, en raison de leur définition volumique [170]. De plus, les hypertextures peuvent être utilisées pour créer des terrains complexes avec des grottes [49] ou des arbres [144]. Afin d'être capable d'effectuer le rendu d'un objet polygonal auquel est appliquée une hypertexture, il est possible de convertir une surface polygonale en représentation pseudo-implicite (en utilisant une structure de squelette). Ceci permet d'appliquer une méthode d'hypertexture en se basant sur des champs de distance [37]. D'autres approches consistent à convertir l'objet en une grille régulière de voxels, construisant ainsi une variation de densité, en se basant sur la distance entre un voxel et un squelette [138].

Toutefois, les méthodes concernant le rendu de textures volumiques nécessitent des systèmes de rendu complexes, comme par exemple le *ray marching*. Ces techniques étant très coûteuses en terme de performance, des travaux se sont concentrés sur le rendu interactif d'hypertextures. Miller [107] propose ainsi une solution utilisant une représentation discrète d'un objet (à base de voxels) stockée sous la forme d'une texture 3D. Le rendu en lui même est effectué en utilisant une technique de lancer de rayon ou un rendu par tranche (voir section 3.4.2). La DMF est ainsi programmée directement dans le *fragment program* associé. Toutefois, ceci implique la création d'un programme spécifique pour chaque hypertexture, ce qui limite actuellement son utilisation à des utilisateurs familiers avec la programmation GPU.

Beaucop de travaux de recherche conernent la définition de fonctions de bruit avec des propriétés statistiques, spectrales et visuelles particulières [120, 91, 92, 169, 122]. Mais la corrélation entre une hypertexture et une DMF basée dur ces bruits n'est pas toujours évidente et nombre de résultats sont obtenus de manière expérimentale. De manière similaire, Dischler propose [38] l'utilisation de courbe de profil afin de créer des fonctions de turbulence avec des propriétés visuelles caractéristiques. Ces fonctions peuvent être étendues en 3D et utilisées comme une DMF. Toutefois, la corrélation avec les champs de distorsion n'est pas intuitive et le contrôle de l'effet visuel final reste ardu.

7.2 Fonction de modulation de la densité (*DMF*)

Dans cette section, nous rappelons brièvement le principe des hypertexures introduit par Perlin et Hoffert [123]. Pour être utilisable dans le contexte des hypertextures, un objet $O(x,y,z)$ doit être défini comme une variation de densité dans l'espace 3D. En règle générale, un volume englobant est défini. En dehors de ce volume, la densité d'un objet est considerée comme nulle, alors qu'à l'intérieur, la densité varie de manière continue entre 0 et 1. Dans ce contexte, une valeur de densité de 1 représente une partie de l'objet totalement opaque et 0 une partie totalement transparente (l'objet n'existe pas en ce point là). Les valeurs entre 0 et 1 sont appelée la région « douce » de l'objet. Le principe majeur des hypertextures consiste à appliquer à cette région douce une ou plusieurs fonctions de modulation de la densité (DMFs). Ces DMFs sont basées sur du bruit, de la turbulence, du biais et du gain. Ces quatre fonctions de base offrent un contrôle sur l'apparence de la DMF finale [123]. Il est à noter toutefois que n'importe quelle définition procédurale peut être utilisée comme une DMF. Un exemple courant est celui de la sphère douce centrée sur l'origine de rayon R :

$$O\big(x,y,z\big) = \begin{cases} 1 - \frac{d}{R} \text{si } (d < R) \\ 0 \text{sinon} \end{cases} \tag{7.1}$$

avec d la distance euclidienne $d = \sqrt{(x^2+y^2+z^2)}$. À l'origine, la densité vaut 1 et décroit vers 0 lorsque la distance à l'origine s'approche de R. Cette sphère douce peut être déformée par une DMF, par exemple basée sur un bruit, de manière suivante :

$$DMF(x,y,z) = O\big(kx,ky,kz\big) \text{ , avec } k = 0.5 + 0.5 \cdot \text{noise}(fx,fy,fz). \tag{7.2}$$

$noise$ est ici une fonction de bruit 3D renvoyant une valeur pseudo-aléatoire entre -1 et 1 et f représente sa fréquence.

Comme on peut le voir sur cet exemple, les hypertextures sont une approche très générale pour ajouter des détails procéduraux à un objet. Toutefois, il n'est pas toujours facile d'utiliser les hypertextures. En effet, cette méthode requiert une définition volumique fonctionnelle d'un objet ainsi qu'une fonction de modulation de la densité. Les surfaces implicites sont très proches d'une représentation volumique fonctionnelle, elles sont souvent utilisées de manière directe avec les hypertextures. En effet, un objet arbitraire peut être converti en un champ de distance où le squelette d'un objet représente le cœur « solide » du modèle, permettant ainsi une utilisation des hypertextures. Toutefois, le rendu des objets défini d'une telle manière reste un problème difficile en temps interactif.

La définition même d'une fonction de modulation de la densité pose également un problème complexe. Premièrement, la déformation, si elle n'est pas définie de manière correcte, peut modifier le volume englobant de l'objet (la frontière au-delà de laquelle la densité est toujours nulle). Ceci implique que des détails « sortent » du volume de définition. Dans le cas d'une méthode de *ray marching* par exemple, l'évaluation du rayon de vue doit commencer à un point donné (de manière intuitive, la densité de l'objet étant nulle en dehors du volume englobant, on commence à sa frontière), ce qui implique qu'il est intéressant d'un point de vue efficacité de connaitre le volume englobant de l'objet. Si des détails se trouvent en dehors de ce volume, la question qui se pose est celle ci : où commencer l'évaluation de la densité le long du rayon de vue pour être sûr de ne pas rater de détails ? Une idée courante consiste souvent à commencer l'évaluation de la densité au plus tôt, c'est-à-dire de commencer à évaluer loin de l'objet. Bien que ceci permette en théorie de ne pas rater de détails, cette idée est inutilisable pour une application interactive. En effet, ceci multiplie les évaluations inutiles (avec une densité de 0), augmentant considérablement le temps de calcul. Dans l'exemple de la sphère douce par exemple, un point P en dehors du rayon initial R peut avoir une densité supérieure à 0 après déformation par la DMF, bien que le volume englobant initial soit une sphère de rayon R. En effet, prenons un point $P = (2R, 2R, 2R)$ (donc en dehors du volume englobant) et une fréquence $f = 1$. Si par hasard, noise$(2R, 2R, 2R) = -1$ alors $k = 0$, ce qui donne $d = 0$ et donc $O(k2R, k2R, k2R) = O(0,0,0) = 1$. Ainsi, la définition de la DMF étant basée sur un bruit, qui est une fonction pseudo-aléatoire, il est difficile de déterminer le nouveau volume englobant d'un objet après déformation. Dans le cas d'un schéma de rendu basé sur le *ray marching*, il devient difficile de choisir un point de départ pour l'évaluation de la densité de l'objet. Toutefois, une solution classique permettant de résoudre ce problème consiste à limiter la gamme d'effet qu'il est possible de représenter en choisissant une DMF qui n'augmente pas le volume englobant de l'objet.

Deuxièmement, il n'existe pas de manière intuitive pour corréler une DMF et un effet visuel. Avec beaucoup d'expérience, il est bien sûr possible de deviner l'effet visuel d'une DMF donnée. La question suivante est encore plus problématique : ayant un certain effet visuel en tête, quelle DMF devrais je programmer pour obtenir exactement ce résultat ?

Nous allons présenter dans la section suivante notre modèle d'hypertextures. Ce modèle étant une spécialisation, il est donc moins générique que le modèle traditionnel des hypertextures. Toutefois, en échangeant la généralité contre une plus grande facilité d'utilisation , nous augmentons le contrôle global des effets visuels sur des objets arbitraires. Du moins, notre approche per-

met un parcours intuitif de solutions sans nécessiter aucune programmation.

7.3 Une nouvelle définition des hypertextures

Cette section décrit notre nouvelle définition pour la modélisation des hypertextures. De la même manière que pour le concept général des hypertextures, un objet $O(x,y,z)$ est ici aussi défini par une densité de variation dans l'espace 3D, c'est-à-dire $O(x,y,z) = d$, avec d un scalaire représentant la densité entre 0 et 1. notre modèle de déformation de l'espace est défini comme suit :

$$\mathrm{DMF}(x,y,z = O(x + k \times \frac{V_x}{\|V\|}, y + k \times \frac{V_y}{\|V\|}, z + k \times \frac{V_z}{\|V\|}) \tag{7.3}$$

avec $V = (V_x, V_y, V_z)$ un champ de vecteurs 3D et $\|\|$la norme du vecteur. k permet d'appliquer la déformation en déplaçant des points le long de V. k peut être considéré comme un facteur de déformation défini comme suit :

$$k = aT\left[\frac{1 + noise(\frac{x+rV_x}{f}, \frac{y+rV_y}{f}, \frac{z+rV_z}{f})}{2}\right] \tag{7.4}$$

a et f représentent respectivement l'amplitude et la fréquence de la déformation. T est ce que l'on appelle une fonction de *transfert de forme*. Cette fonction, retournant une valeur entre 0 et 1, est une table stockée dans une texture sur le GPU. Cette formule est pour le cas 1D ; le cas 2D, résultant en des fonctions plus complexes, sera présenté dans la section 7.3.2. noise représente dans cette formule une fonction de bruit tridimensionnelle renvoyant une valeur pseudo-aléatoire entre -1 et 1. Il est à noter que dans notre modèle, cette fonction de bruit n'est pas évaluée en (x,y,z) mais sur un point déplacé, en fonction du vecteur V et d'un facteur r (une valeur entre 0 et 1). Le champ de vecteurs V et le facteur r permettent à l'utilisateur de contrôler la classe et le type de déformation de l'hypertexture, comme présenté sur la figure 7.2. Avec les hypertextures classiques, la déformation est généralement appliquée le long du gradient de densité de l'objet, ou le long d'un gradient perturbé par une fonction de bruit. Dans l'exemple précédent avec la sphère, la déformation est appliquée le long du rayon. Mais cela consiste en une limitation des possibilités : l'apparition dans notre formulation d'un champ de vecteur V défini de manière explicite améliore les possibilités de création tout en gardant une méthode intuitive (nous discuterons du champ de vecteurs dans une section suivante).

Afin de pouvoir correctement afficher un objet $O(x,y,z)$ ainsi défini, il faut associer à la densité ainsi calculée en chaque point une couleur et une opacité :

nous utilisons ici une fonction de transfert de couleur et d'opacité (une table $RGBA[d]$). Il est à noter que contrairement à d'autres méthodes, nous n'utilisons pas ici directement la valeur de la densité d comme un coefficient d'opacité, mais nous proposons l'utilisation d'une fonction de transfert. Cela nous offre une flexibilité supplémentaire et permet de découpler la transparence de la densité définissant un objet volumique. Par exemple, il devient possible d'appliquer un schéma d'hypertexture sur un objet totalement opaque. Un tel objet serait défini par une variation de densité entre 0 et 1 mais aurait une opacité de 1. Afin de restituer le phénomène de l'illumination de manière correcte, le gradient ∇O peut être calculé par dérivation discrète (voir section 7.4 pour plus de précision).

Nous nous concentrons maintenant sur l'influence des paramètres de ce modèle, principalement celle du champ de vecteurs et de la fonction de transfert de forme qui, à première vue, semblent restreindre les variétés de DMF par rapport à une définition totalement procédurale. Les paramètres a et f sont des paramètres classiques : le premier coefficient a représente la « profondeur » de l'hypertexture, c'est-à-dire l'amplitude de la déformation. La valeur $a = 0$ correspond à une profondeur nulle, donc à une déformation nulle. Le second coefficient f permet de déterminer la « fréquence » et donc ainsi la taille des éléments de l'hypertexture. Plus f est grand, plus les éléments d'hypertextures sont présents par unité de volume et plus ils sont petits. La figure 7.9 illustre l'effet de ces coefficients.

La fonction de transfert de forme représente le cœur de notre modèle. Elle est représentée comme une table T à une ou deux dimensions. Elle permet le contrôle de la forme de la déformation et donc de l'apparence visuelle finale des éléments ajoutés à l'objet. Cette fonction est indexée par une fonction de bruit de manière à maintenir la nature procédurale de la déformation. Ces fonctions peuvent être éditées en temps réel en utilisant les outils adéquats. Ces outils peuvent être externes ou intégrés à l'application. Les exemples de ce chapitre ont été réalisés avec des outils d'édition de courbes et un simple outil de dessin. Une fois ces fonctions réalisées (ou modifiées), elles sont instantanément transmises à la carte graphique, montrant ainsi en temps réel l'effet visuel sur l'objet hypertexturé. Nous allons maintenant discuter dans les prochaines sections les cas des fonctions de transfert 1D et 2D ainsi que l'influence du champ de vecteurs.

7.3.1 Fonction de transfert de forme 1D

Comme introduit par la formule précédente 7.4, la fonction de bruit (noise) est utilisée comme un index par la fonction de transfert de forme. Ainsi, l'aspect sous-jacent de la fonction de bruit joue un rôle important pour l'effet visuel de la fonction de transfert. De manière plus spécifique, la fréquence et la distribution spatiale des valeurs du bruit ont un impact important sur le résultat final. La forme de base d'une fonction de bruit de Perlin traditionnelle ainsi que l'histogramme de la densité de probabilité sont représentés sur la figure 7.3. Nous pouvons voir que les valeurs proches de 0 sont plus fréquentes que les valeurs extrêmes -1 et 1. Il s'agit en fait d'une ditribution normale centrée en 0. La partie droite de cette figure montre la même fonction de bruit, mais au lieu d'utiliser la valeur du bruit, nous affichons la fréquence correspondante. Sur cette image, l'isoligne 0 est bien mise en évidence.

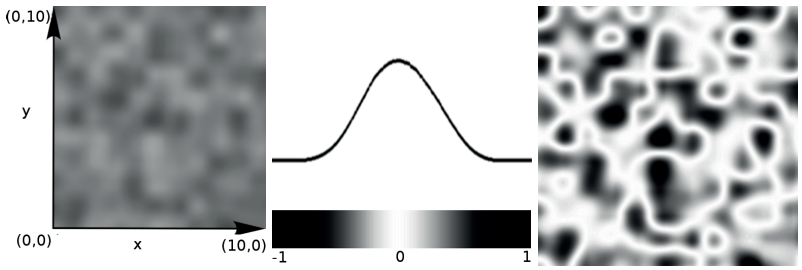


FIGURE 7.3 – Une coupe 2D d'une fonction de bruit de Perlin 3D ainsi que son histogramme de densité de probabilité.

La figure 7.4 montre les résultats obtenus pour différentes fonctions de transfert de forme 1D (représentées dans la première colonne). Nous avons appliqué ces fonctions à une simple sphère douce avec une densité décroissante au fur et à mesure que la distance au centre de la sphère s'accroit. Pour une meille compréhension visuelle, nous avons utilisé une fonction escalier pour l'opacité, ce qui résulte en l'extraction de l'isosurface sans effets de semi-transparence.

La seconde colonne montre les résultats obtenus en indexant la fonction de transfert de forme avec la fonction de bruit de la figure 7.3 (nous rappelons ici que nous montrons une coupe 2D d'une fonction 3D). La colonne suivante montre les résultats lorsque la fonction de transfert de forme est appliquée à la sphère douce avec un facteur $r = 0.0$, $r = 0.5$ et $r = 1.0$, en accord avec le modèle précédemment décrit (la deuxième image est obtenue en utilisant un plan de coupe, permettant ainsi l'affichage de l'intérieur de la sphère).

Pour ces exemples, nous avons utilisé un champ de vecteurs projetant les

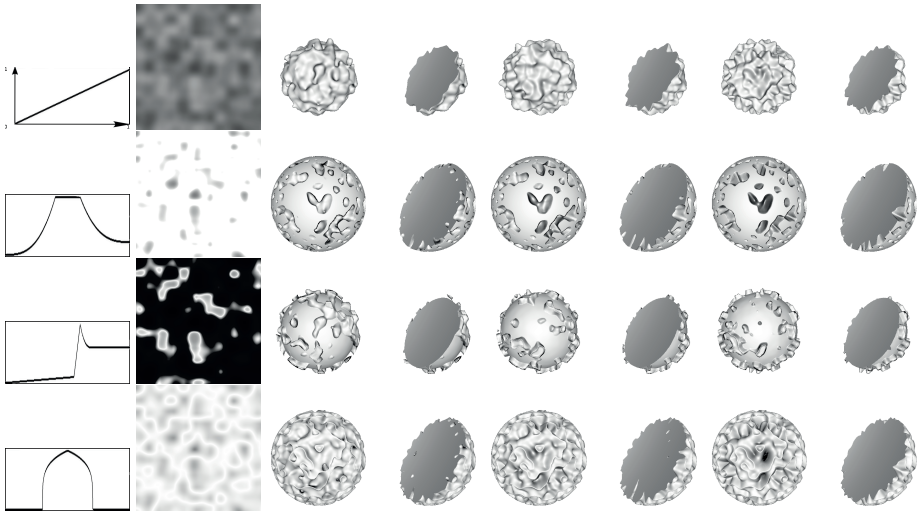


FIGURE 7.4 – Exemples de fonction de transfert de forme 1D et les résultats correspondant sur la sphère douce pour les valeurs $r = 0$, $r = 0.5$ et $r = 1$ de gauche à droite.

points sur le point de la surface le plus proche (d'autres champs de vecteurs seront présentés plus tard). Le cas $r = 0$ signifie que la fonction de bruit 3D est évaluée exactement aux coordonnées du point considéré sans aucun décalage. En prenant en compte le champ de vecteurs et l'amplitude de la déformation, cela peut créer des effets complexes, comme des cavités.

La figure 7.3 nous aide à comprendre les résultats de la figure 7.4. Principalement, elle aide à la prédiction de l'effet d'une fonction de transfert 1D sur le résultat visuel. Le premier exemple de la figure 7.4, sur la première ligne, montre une fonction de transfert identité, c'est-à-dire $T[x] = x$. Il en résulte une déformation basée sur une fonction de bruit traditionnelle. Cette image peut être utilisée comme référence, montrant ainsi la déformation générée lorsqu'aucune fonction de transfert n'est utilisée. Sur le second exemple, à la deuxième ligne, la fonction de transfert a une valeur de 1 partout, sauf aux deux extrémités, où la valeur décroit lentement pour former des cavités.

La troisième ligne montre une fonction de transfert qui crée des bosses avec un profil ressemblant à un cratère lorsque la valeur du bruit atteint 1. Une fois encore, le résultat obtenu sur la sphère pour toutes les valeurs de r est assez intuitif. Le dernier exemple montre quant à lui une fonction de transfert qui crée des bosses lorsque la valeur du bruit atteint 0. Il en résulte la création d'isolignes correspondant à la valeur de bruit 0.

La dernière colonne de la figure 7.4 illustre le cas extrême où $r = 1.0$ et mérite plus ample discussion. Comme le champ de vecteurs que nous utilisons

dans ce cas pointe sur le point le plus proche à la surface de la sphère (c'est ce point qui sera évalué par la fonction de bruit), tous les points le long d'un rayon de la sphère auront le même facteur de déplacement. De plus, dans la sphère, le rayon correspond à la direction de la normale à la surface (i.e. le gradient). L'ensemble de ces remarques donne un résultat visuel final qui ressemble à un *displacement mapping* traditionnel.
Toutefois, notre approche diffère du *displacement mapping* en plusieurs points : au lieu d'utiliser une paramétrisation de la surface parfois difficile à obtenir pour des surfaces arbitraires, notre approche est basée sur un champ de vecteurs qui, dans ce cas particulier, pointe vers la surface. Un tel vecteur est non seulement plus facile à calculer qu'une paramétrisation, mais peut aussi être modifié, ce qui offre la possibilité de créer des effets visuels plus complexes, comme des cavités, qui bien que fréquentes dans le monde réel, ne peuvent être modélisées avec une simple technique de *displacement map.* Même avec une fonction de transfert 1D, l'utilisateur a ici la possibilité d'obtenir des cavités plus ou moins importantes en modifiant le facteur r.

Comme illustré dans ces exemples, les fonctions de transfert 1D représentent un outil intuitif et efficace pour contrôler l'aspect visuel final. Toutefois, la création d'hypertextures plus complexes reste limitée. Il est possible de créer uniquement des structures symétriques très simples comme des bulles, des bosses, des pics, ou des trous. De manière à créer des détails de surface plus complexes, nous pouvons utiliser une combinaison de plusieurs fonctions de bruit, avec pour chacune une fonction de transfert de forme particulière, ainsi que différents paramètres d'amplitude et de fréquence.

La figure 7.5 montre trois exemples de structures complexes obtenues en utilisant une combinaison de plusieurs fonctions de transfert et de fonctions de bruit. Pour cela, nous remplaçons la formule précédente 7.4 par la suivante :

$$k = \oplus_{i=1}^{n} a_i T_i \left[\frac{1 + noise(\frac{x+r_i V_x}{f_i}, \frac{y+r_i V_y}{f_i}, \frac{z+r_i V_z}{f_i})}{2} \right] \tag{7.5}$$

$\oplus$ représente ici un opérateur arbitraire choisi parmi un ensemble d'opérateurs prédéfinis (dans notre cas, nous avons défini deux opérateurs uniquement : la somme et le mélange). Pour réaliser le premier exemple, nous avons utilisé une somme de trois bruits avec la même fonction de transfert (la seconde fonction de transfert de la figure 7.4, c'est-à-dire celle représentant des cavités). Seule l'amplitude et la fréquence sont différentes pour chaque fonction de bruit, créant ainsi des trous de tailles différentes. Le second exemple quant à lui, a été réalisé au travers d'une somme de cinq bruits, avec deux fonctions de

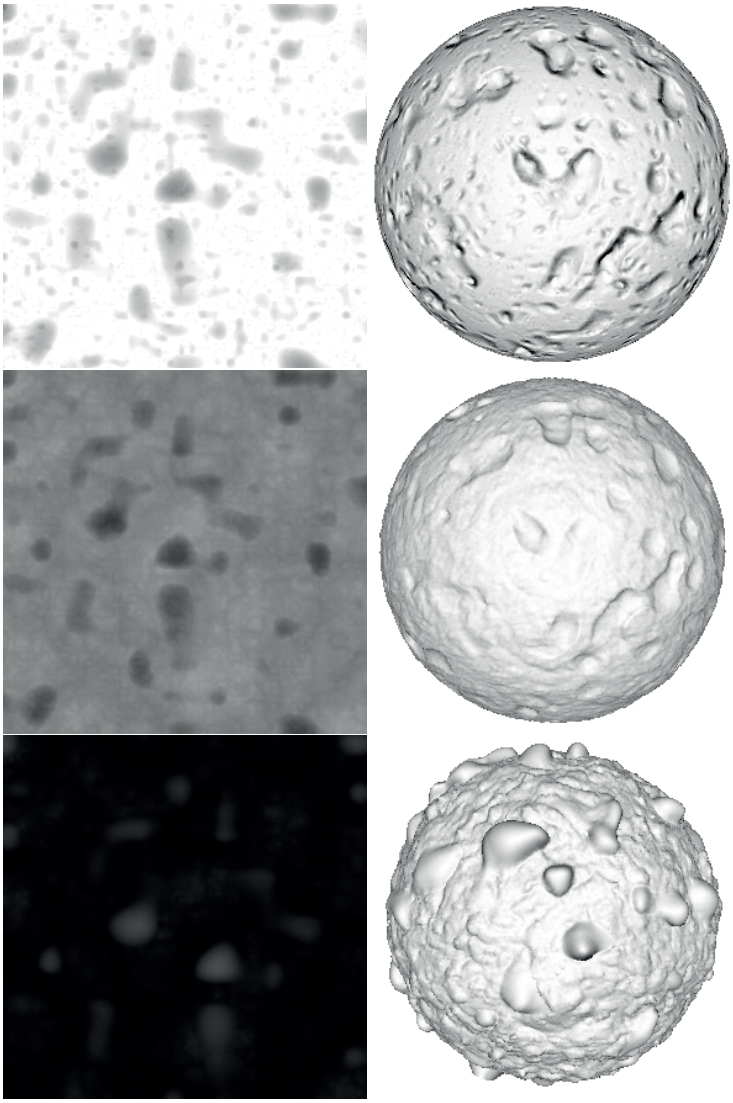


FIGURE 7.5 – Une combinaison de plusieurs fonctions de transfert de forme avec différentes fonctions de bruit à différentes échelles ($r = 0.5$).

transfert différentes. La première fonction de bruit utilise à nouveau la fonction de transfert en forme de trous tandis que les quatres fonctions suivantes ont une forme de V ($T[x] = abs(x)$) avec une fréquence croissante, créant ainsi une turbulence. Le dernier exemple montre à nouveau une somme de cinq bruits, les quatre derniers utilisant une fonction de transfert en forme de V (générant ainsi une turbulence). La première fonction de bruit indexe cette fois une fonction de transfert en forme de bosses. Dans ce cas toutefois, nous n'utilisons pas l'opérateur somme pour le premier bruit mais l'opérateur de mélange. Ceci se

traduit par le fait que les bosses ont une apparence lisse et ne sont pas affectées par la turbulence.

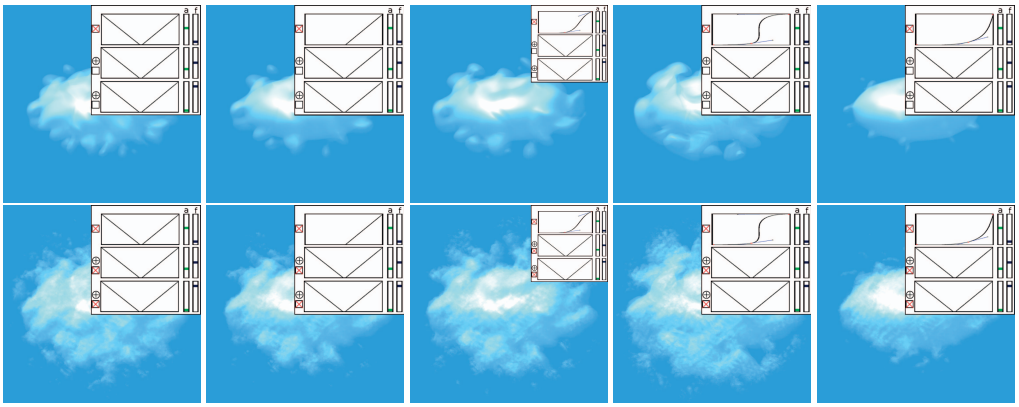


FIGURE 7.6 – Un exemple d'interaction. Les six images illustrent l'édition progressive de la fonction de transfert de forme. La première ligne montre le modèle déformé en utilisant une seule fonction, ainsi que la fonction de transfert correspondante (18 fps). La deuxième montre l'effet obtenu en activant deux fonctions supplémentaires avec une échelle plus petite et une plus haute fréquence (7.1 fps).

La figure 7.6 montre un autre exemple d'utilisation de plusieurs fonctions de bruit (dans ce cas précis, trois). Elle illustre comme la meso-structure évolue lorsque l'on modifie progressivement la fonction de transfert de forme. Nous présentons également un exemple simple d'éditeur basé sur des *splines*. Pour cet exemple, l'objet de base est une ellipse. Nous avons utilisé une fonction d'opacité linéaire pour obtenir de la semi-transparence. L'amplitude a été définie en fonction de la taille de l'ellipse pour créer des déformations importantes. Le coefficient r a été fixé à 0 pour autoriser des changements topologiques importants. La première ligne de cette figure montre le résultat lorsqu'une seule fonction de bruit est utilisée. Ceci montre qu'il est intuitif de définir des détails de mésostructure à différentes échelles.

La combinaison de différentes fonctions de bruit et de fonctions de transfert de forme permet d'augmenter la variété des détails de meso-structure sur une surface, mais un contrôle plus précis peut encore être obtenu. Nous proposons pour cela de discuter de l'augmentation de la dimension de la fonction de transfert, en ajoutant pour cela un deuxième paramètre. Ce second paramètre, utilisé lui aussi pour indexer la fonction de transfert de forme, ne dépend pas cette fois d'une fonction de bruit mais de l'objet en lui-même.

7.3.2 Fonction de transfert de forme de dimension supérieure

Utiliser une fonction de transfert de forme à deux dimensions consiste simplement à utiliser une table 2D à la place d'une table 1D. Pour une table 2D, nous avons toutefois besoin d'un second paramètre. Ce paramètre peut être lié à l'objet lui-même, comme la densité par exemple. Nous remplaçons le calcul précédent du facteur k de la formule 7.4 par la nouvelle formule suivante :

$$k = aT_{2D}\left[\frac{1 + noise(\frac{x+rV_x}{f}, \frac{y+rV_y}{f}, \frac{z+rV_z}{f})}{2}, O(x,y,z)\right] \tag{7.6}$$

La figure 7.7 montre quelques exemples de fonctions de transfert de forme 2D appliqués à la sphère douce. Dans ce cas, le deuxième paramètre est la densité de la sphère douce, comme décrit dans la formule 7.6. La densité s'accroit alors que l'on se déplace vers le centre de la sphère. La première ligne de la figure peut être utilisée comme image de référence : on utilise une fonction de transfert 2D constante qui ne varie pas avec la densité de l'objet. C'est donc l'équivalent d'une fonction de tranfert 1D. Le coefficient r a été fixé à 1 de manière à mettre en évidence la variation le long du rayon de la sphère (avec une fonction de transfert 1D et $r = 1$, il n'y a pas de variation le long du rayon). Des structures complexes peuvent désormais être définies, comme les structures en forme de champignon du dernier exemple. Le contrôle du résultat final reste néamoins intuitif (la corrélation avec la fonction de transfert 2D est triviale).

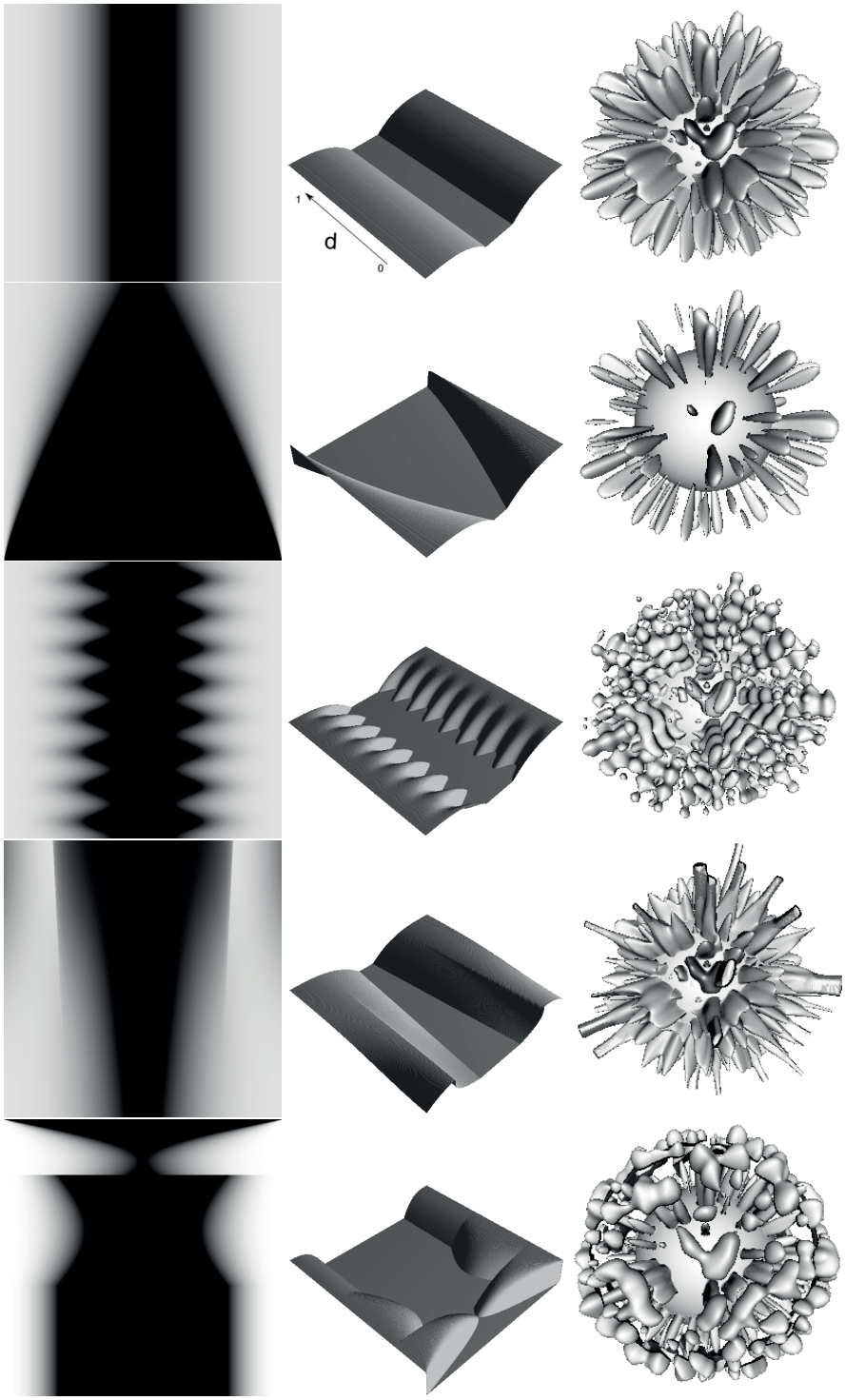


FIGURE 7.7 – Exemples de fonctions de transfert de forme 2D afin d'obtenir des variations en fonction de la profondeur.

À l'instar des fonctions de transfert 1D, des combinaisons de plusieurs bruits peuvent être utilisées pour augmenter les possiblités de déformations. Il est également possible de définir des fonctions de transfert de forme de dimensions supérieures. Toutefois, de telles fonctions risquent d'être moins intuitives et plus difficiles à manipuler. Premièrement, des fonctions de transfert de dimension supérieure requièrent de trouver des paramètres d'indexation ayant un sens pour permettre une modélisation intuitive. De plus, modéliser une fonction de transfert 3D en temps réel et la charger sur la carte graphique peut être en soi un problème difficile.

7.3.3 Champ de vecteurs

Avec les hypertextures traditionnelles, la déformation est généralement appliquée dans la direction du gradient de la variation de densité ou le long d'un gradient perturbé par un bruit (cette méthode a été utilisée par Perlin et Hoffert [123] pour obtenir des cheveux courbés par exemples).
Dans notre cas, nous utilisons un champ de vecteurs explicite V. Il a deux objectifs :

- Premièrement, à l'instar des hypertextures classiques, il définit une direction au moyen d'un vecteur unitaire $\frac{V}{\|V\|}$ qui est utilisé pour déformer l'espace en déplacant des points le long de cette direction. Le champ de vecteurs permet de contrôler de manière efficace l'orientation de la distortion . De plus, il nous permet de prédire comment la frontière de l'objet sera étendue dans le cas extrême. En effet, puisque la fonction de transfert de forme T renvoie une valeur entre 0 et 1, le facteur d'amplitude a représente la valeur de déplacement maximale. C'est-à-dire qu'un point P est déplacé au maximum de $P + \frac{V}{\|V\|}$. Si nécessaire, cela nous permet de calculer une nouvelle frontière pour l'objet, c'est-à-dire une frontière en dehors de laquelle la densité est toujours nulle. Ceci permet d'augmenter l'efficacité des algorithmes de lancer de rayon par exemple. Il est à noter toutefois que, dans notre cas, nous orientons la déformation vers l'intérieur afin de limiter l'expansion de l'objet.
- Deuxièmement, le champ de vecteurs est utilisé pour déterminer la localisation exacte à laquelle la fonction de bruit est evaluée, c'est-à-dire en $P + r\,V$. Dans le cas de la sphère douce, nous avons par exemple défini un champ qui déplace tous les points de l'espace exactement sur la surface de la sphère le long de la ligne passant par son centre (donc le point le plus proche sur la surface). Ce cas est à nouveau illustré par la figure 7.8. La forme de la déformation résultante est corrélée à la fonction de transfert de forme (dans ce cas elle crée des pics). Le facteur r contrôle la distance à laquelle les points sont déplacés pour évaluer la fonction de bruit.

Quand r a pour valeur 0, aucun déplacement n'est appliqué et la fonction de bruit est évaluée aux coordonnées du point d'origine. Lorsque r vaut 1, un déplacement maximal est au contraire appliqué.

Contrairement au *displacement mapping* classique, où seuls les points sur la surface sont déplacés dans la direction de la normale, les hypertextures offrent plus de flexibilité, comme illustré par la figure 7.8. Le premier exemple (en haut à gauche) montre le cas décrit précédemment : un champ de vecteurs qui projette les points sur la surface de la sphère le long du rayon. L'exemple de la seconde ligne (à gauche) illustre un champ de vecteurs projetant les points dans la direction opposée d'un point quelconque de la sphère. L'exemple de la troisième ligne à gauche montre quant à lui un champ de vecteurs avec une direction constante verticale. En haut à droite de la figure, on peut observer un champ de vecteurs pointant en direction du bord de la sphère mais perturbé par un champ de vecteurs basé sur un bruit. Cela donne des trajectoires plus aléatoires, et peut même créer des structures de branches. L'exemple suivant illustre un champ curviligne tournant autour d'un axe vertical. Finalement, le dernier exemple représente le champ de vecteurs de l'exemple précédent mais avec une orientation et une magnitude perturbée par un bruit. Comme le montre cette figure, notre définition du champ de vecteurs, qui peut non seulement suivre le gradient et la direction de la normale à la surface mais aussi être perturbé par un bruit, augmente de manière significative les possiblités de modélisation. En outre, les perturbations aléatoires des champs de vecteurs peuvent être intégrées au *fragment program* de manière triviale, gardant ainsi la nature procédurale du modèle.

7.4 Utilisation de notre méthode de rendu

Nous avons vu dans les sections précédentes une nouvelle définition du modèle d'hypertextures. Cette définition permet l'ajout de détails procéduraux pour des objets volumiques en temps interactif en utilisant notre méthode d'ajout de détails volumiques décrite au chapitre 4. Son principe est d'utiliser une des deux méthodes de rendu (pour un objet défini de manière volumique ou surfacique) pour générer des segments de vue intersectant l'objet et d'évaluer la densité de l'objet aux échantillons placés le long de ces segments avec notre modèle d'hypertexture. Le nombre d'échantillons placés le long d'un segment de vue est contrôlé dans cette application par un paramètre s. Nous avons défini ici s comme étant la distance entre deux échantillons successifs le long d'un segment $[p_e, p_s]$. De plus amples discussions sur ce paramètre seront présentées dans la section 7.5.

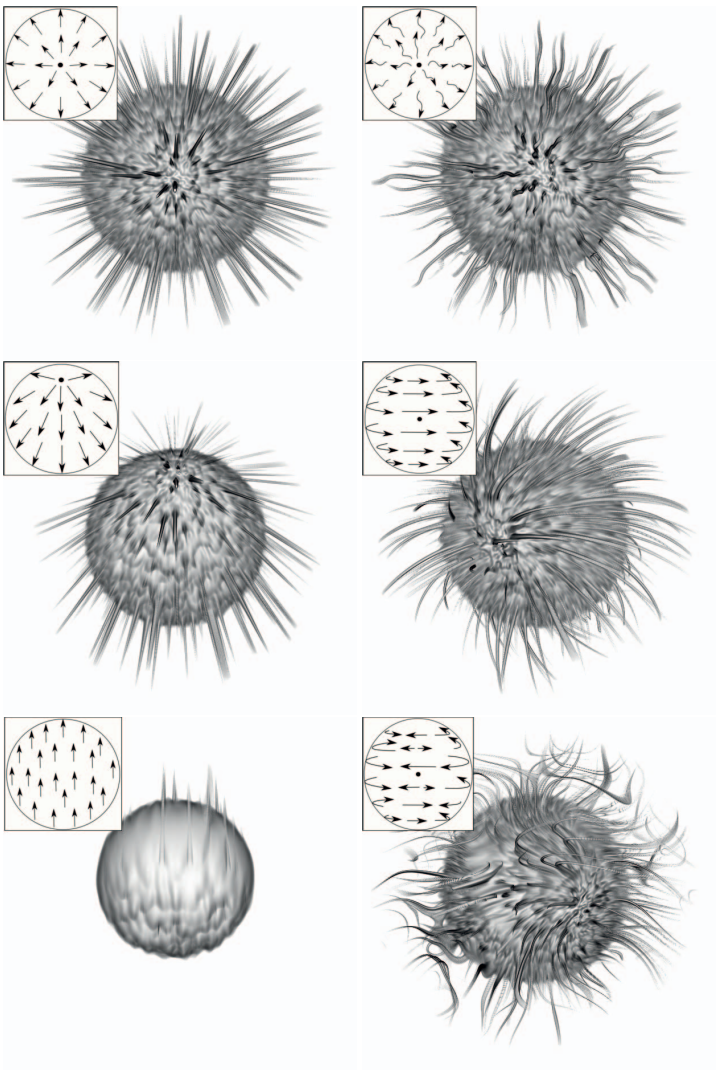


FIGURE 7.8 – Influence du champ de vecteurs. Ce dernier peut de plus être perturbé par un vecteur de bruit pour permettre l'obtention d'effets plus complexes.

Dans le cas de modèles définis de manière volumique, nous utilisons une représentation basée sur un ensemble d'échantillons associés à un noyau de reconstruction de taille variable. Une étape de précalcul permet de générer pour n'importe quel modèle 3D un tel ensemble. De plus, un tel modèle peut être optimisé de manière à diminuer le nombre d'échantillons nécessaires en faisant varier la taille des noyaux de reconstruction. En effet, ce problème consiste à approximer un volume par une union de sphères de taille variable.

Dans le cas de modèles définis de manière surfacique, il s'agit d'utiliser une

représentation de la frontière de l'objet comme support pour générer des segments. Dans ce cas néanmoins, il est nécessaire de connaitre pour chaque point à l'intérieur du modèle une estimation de sa densité. En pratique, nous utiliserons une texture 3D grossière associant à chaque texel une densité. Cela est suffisant pour obtenir en chaque point de l'espace 3D à l'intérieur du modèle une estimation de sa densité.

Pour ces deux méthodes de rendu, il est à noter que nous utilisons une technique d'accélération basées sur l'opacité. En effet, nous utilisons dans les deux cas une méthode traitant les échantillons du plus proche au plus lointain. Cela nous permet d'éliminer immédiatement les calculs pour un segment situé sur un rayon de vue émanant d'un fragment qui à déjà une opacité cumulée maximale (dans ce cas, les éléments suivants seront masqués) ce qui permet d'éliminer les calculs inutiles. Cette optimisation est naturellement moins efficace pour des objets ou phénomènes hautement transparents, ce qui se traduit généralement par des performances amoindries.

Afin d'obtenir des images réalistes, il convient pour la plupart des modèles d'utiliser un schéma d'illumination efficace. Dans cette application, notre technique d'illumination est basée sur un modèle de Phong. Celui-ci requiert une normale en chaque point, c'est-à-dire ici le gradient de la densité. Toutefois, le gradient de la déformation introduite par notre hypertexture procédurale n'est pas connu à l'avance et ne peut pas être précalculé. De ce fait, nous avons à le calculer par dérivation discrète lors du rendu. L'utilisation des différences centrales requiert toutefois six évaluations supplémentaires [1] et a un impact non négligeable sur les performances de rendu.

Notre méthode de rendu est de plus compatible avec une implémentation de l'ombrage. En effet, dans le cas de petits détails ajoutés par hypertextures, où des bosses, des creux et des pics sont crées, les ombres jouent une part importante pour la qualité finale du résultat. Pour introduire des ombres, une approche reposant sur un précalcul est bien entendu impossible pour une méthode procédurale. Nous avons choisi d'implémenter ici les ombres en nous basant sur une approche de type lancer de rayon. Pour chaque échantillon, un rayon d'ombre est généré en direction de la source de lumière (ponctuelle). Ce rayon est échantilloné, permettant ainsi l'évaluation de notre modèle d'hypertexture le long du rayon. Les échantillons d'origine sont ombrés en fonction du résultat de l'évaluation des échantillons d'ombres. Néanmoins, un schéma

1. Il est possible d'optimiser cette technique au sein du *fragment program* afin de devoir calculer au plus 5 évaluations dans la plupart des cas.

d'échantillonnage régulier conduirait à évaluer des échantillons d'ombre se situant loin de l'échantillon de matière à ombrer, ce qui conduirait à une perte de performances. Nous résolvons ce problème en utilisant une technique d'échantillonnage proche de celle des *vicinity shadows*[143]. C'est-à-dire que nous choisissons de restreindre les échantillons d'ombre générés à un périmètre proche de l'échantillon d'origine. Cela revient à négliger les ombres des éléments éloignés du point à ombrer. En pratique, l'impact visuel de l'ombre d'un élément très éloigné de l'échantillon à ombrer reste minime. Ceci nous permet d'obtenir des ombres précises pour les éléments proches tout en gardant des performances interactives. De plus, il est possible d'utiliser une des techniques d'ombrage interactive connue, tel les *shadow maps*[166] de manière à approximer l'ombrage de la macrostructure sur elle-même.

En ce qui concerne l'évaluation d'une fonction de bruit réalisée sur la carte graphique, nous utilisons une implémentation matérielle du *simplex noise* de Perlin [58, 121]. Grâce à la définition compacte des hypertextures (liée à la nature procédurale du bruit), l'utilisation de la mémoire graphique est assez limitée[2]. De plus, il est seulement nécéssaire de stocker en mémoire graphique le champ de vecteurs et les différentes fonctions de transferts (des textures 1D ou 2D). Le champ de vecteurs est en outre utilisé pour indiquer une direction de déplacement grossière. Il n'est donc pas nécéssaire de le définir dans une résolution élevée. Il est même possible, pour les champ de vecteurs les plus simples (une direction uniforme perturbée ou non par un bruit) de le calculer au vol. En pratique, nous définissons le champ de hauteur comme les canaux RVB d'une texture 3D basse résolution (pour tous les exemples présentés dans ce chapitre, une texture de résolution 64^3 est utilisée). Finalement, la densité, qui varie elle aussi de manière grossière, est stockée dans le canal alpha de cette texture 3D.

7.5 Résultats

Nous présentons dans cette section les résultats de cette application de notre méthode de rendu. Nos tests de performances ont été réalisés sur un Intel Core2 Quad avec comme carte graphique une GeForce GTX 280. Á moins que ce ne soit précisé différement, les performances ont étés mesurées sur une fenêtre de taille 512×512 avec le modèle couvrant approximativement 75 % de la fenêtre. Les divers paramètres de notre formulation permettent un contrôle

2. Le modèle en lui même est défini comme une collection de sommets 3D et peut être stocké efficacement dans un *Vertex Buffer Object*.

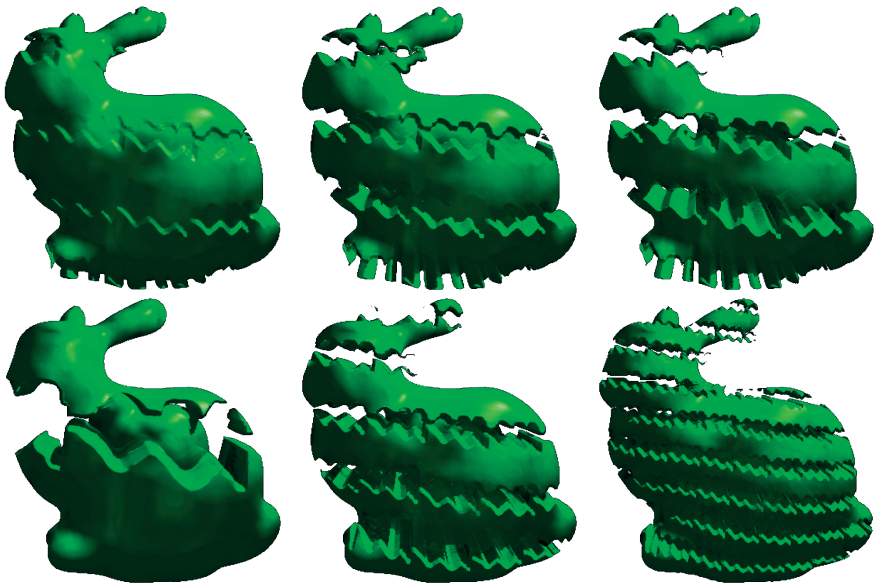


FIGURE 7.9 – Influence des paramètres d'amplitude (haut) et de fréquence (bas).

précis et efficace du résultat.

Premièrement, les paramètres de fréquence f et d'amplitude a sont classiques et intuitifs. Leurs effets sur une déformation sont présentés sur la figure 7.9. Ils permettent de contrôler la « taille » des éléments de meso-structure ainsi que leur répétition. Le paramètre s (le pas d'échantillonnage) affecte directement la qualité et les performances. Il permet à l'utilisateur de choisir entre une exploration interactive des solutions et un rendu de haute qualité. Les première et seconde colonnes de la figure 7.10 montrent pour les deux méthodes de rendu une estimation visuelle des différences entre une valeur de s dédiée à la production d'image de haute qualité (HQ) ($s = 0.002$) et une valeur orientée exploration interactive (IE)($s = 0.015$). Comme prédit dans le chapitre 5, le méthode de rendu d'objet volumique (en haut), bien que plus lente, propose des résultats plus lisses, en particulier avec un pas d'échantillonnage identique, comme illustré dans la dernière colonne. Cela est encore une fois dû à la génération de plusieurs segments le long d'un même rayon de vue.

La table de la figure 7.11 montre les performances obtenues avec différents paramètres sur des objets avec la même hypertexture en utilisant les deux méthodes de rendu et faisant varier le taux d'échantillonnage, la méthode d'ombrage ainsi que l'ajout ou non d'ombres portées. Comme cette table l'illustre, les performances de l'application décroissent avec l'introduction de l'ombrage de Phong. Ceci est principalement dû à l'introduction du gradient, calculé ici

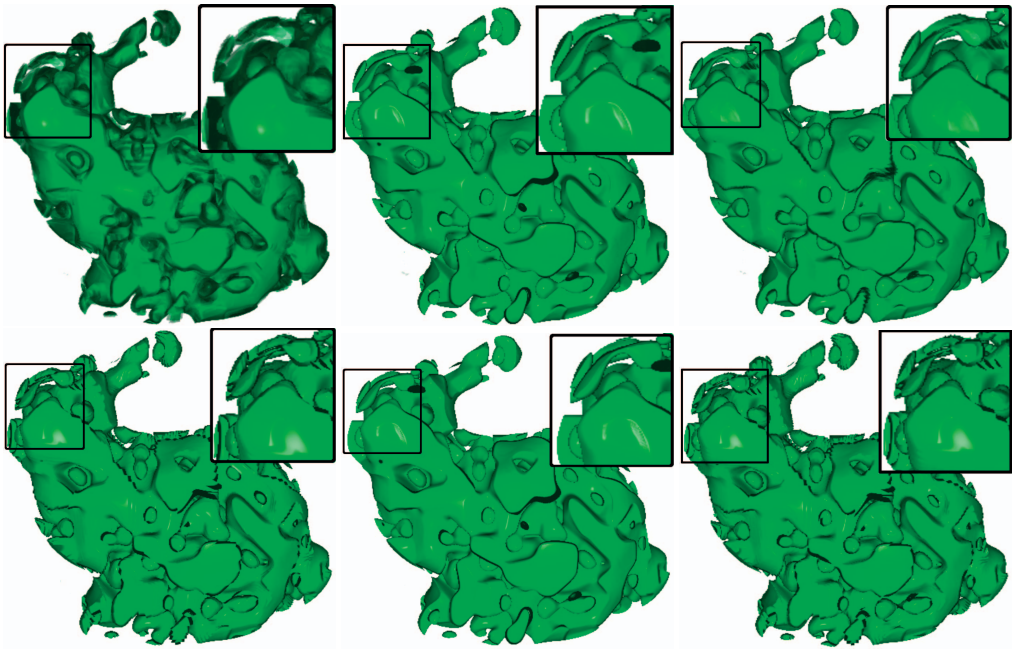


FIGURE 7.10 – Comparaison visuelle entre la méthode volumique (ligne du haut) et la méthode surfacique (ligne du bas). La première et la seconde colonne correspondent respectivement à l'exploration interactive (IE, en haut : 20 fps, en bas : 50 fps) et à la haute qualité (HQ, en haut : 8 fps, en bas : 12 fps). La dernière colonne montre les résultats avec le même pas d'échantillonnage pour les deux méthodes (en haut : 12 fps, en bas : 60 fps).

par différences centrales, qui requiert 6 évaluations supplémentaires de notre formule d'hypertexture pour chaque échantillon.

Comme décrit dans le chapitre introduisant notre méthode de rendu (chapitre 4) cette table montre que les performances de cette application sont plus liées au pas d'échantillonnage qu'au nombre de primitives composant chaque objet. Ceci illustre encore une fois la forte dépendance de notre méthode de rendu de meso-structure entre l'apparence de la meso-structure et les performances, (presque) indépendamment de la complexité de la macro-structure de l'objet.

En ce qui concerne le coût de l'évaluation d'un échantillon, il est fortement lié dans cette application au calcul de la fonction de bruit. Bien que l'implémentation matérielle que nous avons utilisé est performante [3]., nous avons pu estimer que l'évaluation de la fonction de bruit introduit une baisse de performance de l'ordre de 65%. En utilisant une combinaison de trois bruits, plus de

3. Nous avons utilisé l'implémentation du *simplex noise* proposée par Gustavson.[58].

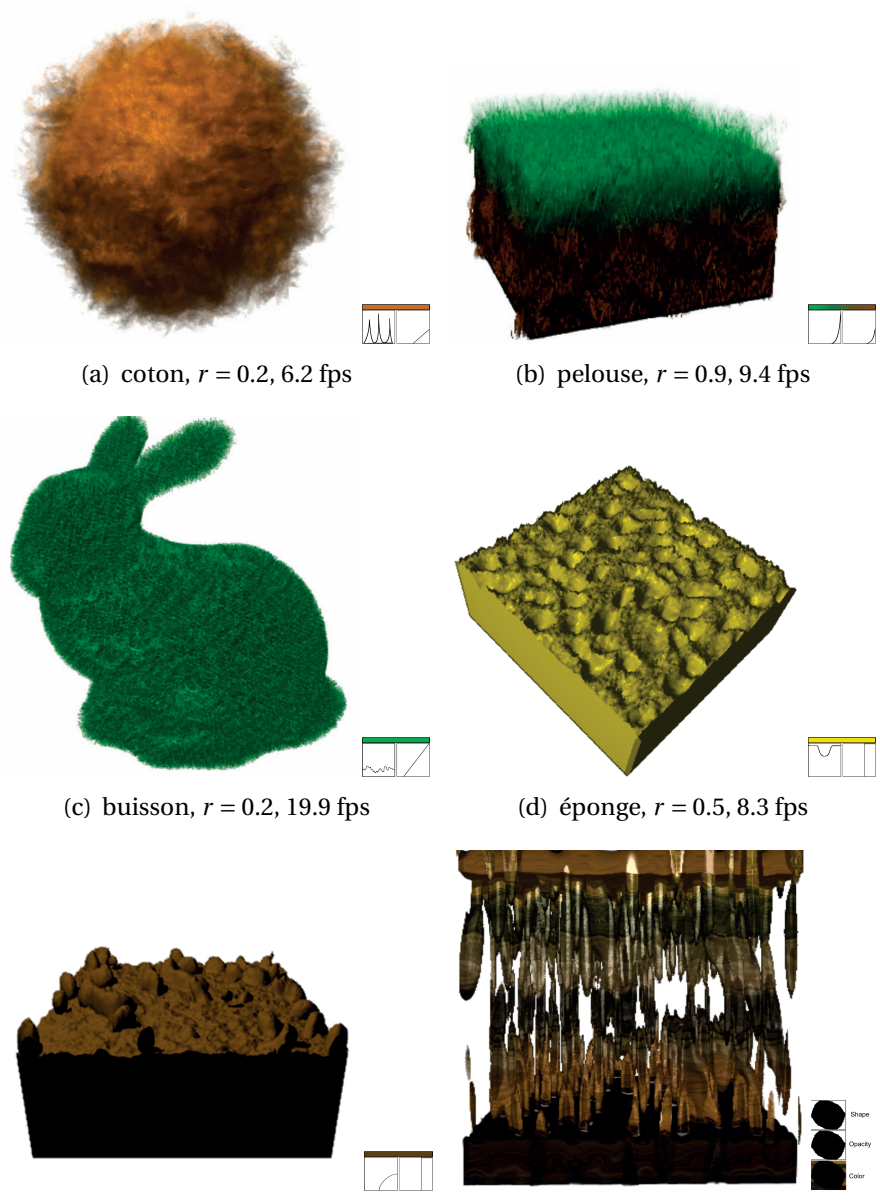
		Bunny (7608 splats)		Teapot (4483 splats)	
		HQ	IE	HQ	IE
Volumique	Sans Shading	12.8 fps	30.0 fps	15.3 fps	35.0 fps
	Phong	8.5 fps	20.8 fps	9.5 fps	20.0 fps
	Phong et ombres	6.3 fps	14.5 fps	8.7 fps	19.1 fps
Surfacique	Sans Shading	12.8 fps	> 60.0 fps	17.0 fps	> 60.0 fps
	Phong	12.2 fps	51.0 fps	15.1 fps	60.0 fps
	Phong et ombres	9.0 fps	32.4 fps	12.1 fps	58.1 fps

FIGURE 7.11 – Comparaison des méthodes pour un objet surfacique ou volumique. Les performances dépendent fortement de l'échantillonnage et varient avec la qualité. Les différences visuelles correspondant à la haute qualité (HQ) ou à la visualisation interactive (IE) sont présentés sur la figure7.10.

85% du temps de calcul est dédié au calcul de cette fonction de bruit. Néanmoins, bien que ces temps de calcul soient nécessaires dans cette application afin de permettre l'édition dynamique des hypertextures, il est envisageable de concevoir une application se reposant sur un précalcul afin d'accroître les performances. Ceci conviendrait à une application se concentrant sur le rendu de mésostructures à hautes performances. Toutefois, une telle application ne se reposerait plus sur des principes procéduraux et arborerait à nouveau les défauts de méthodes reposant sur une définition explicite de la meso-structure.

Afin d'évaluer l'apport des fonctions de transfert, particulièrement les fonctions d'opacité et de forme, à la modélisation et à la reproduction d'effets naturels, nous avons conduit une série de tests sur des utilisateurs familiers avec notre système. Les utilisateurs ont eu la possibilité d'éditer et de modifier les fonctions de transfert ainsi que de changer les paramètres d'amplitude a, de fréquence f et r. Le but était de reproduire certains des effets naturels illustrés dans la figure 7.1. Les résultats obtenus après environ 20 minutes de manipulation sont montrés sur la figure 7.12. Comme le souligne cette figure, les expériences montrent que ces fonctions de transfert peuvent aider à la reproduction de phénomène naturels sans nécessiter la connaissance préalable d'un langage de programmation. La laine, la pelouse et le buisson ont été obtenus avec une fonction de transfert de forme 1D. L'éponge et les mottes de terre ont été obtenus avec trois bruits. Finalement, pour reproduire la grotte, nous avons utilisé le modèle du cube en conjonction avec des fonctions de transferts 2D.

Des résultats supplémentaires ont étés crées et sont appliqués sur différents modèles (figure 7.13). Finalement, la figure 7.14 montre qu'une fois qu'une hypertexture à été éditée , elle peut être aisément être utilisée dans un système de



(a) coton, $r = 0.2$, 6.2 fps (b) pelouse, $r = 0.9$, 9.4 fps

(c) buisson, $r = 0.2$, 19.9 fps (d) éponge, $r = 0.5$, 8.3 fps

(e) mottes de terre, $r = 0.6$, 8.0 fps (f) grotte, $r = 0.85$, 6.8 fps

FIGURE 7.12 – Résultats d'une expérience concernant la capacité à reproduire un phénomène naturel en laissant les utilisateurs éditer les fonctions de transfert.

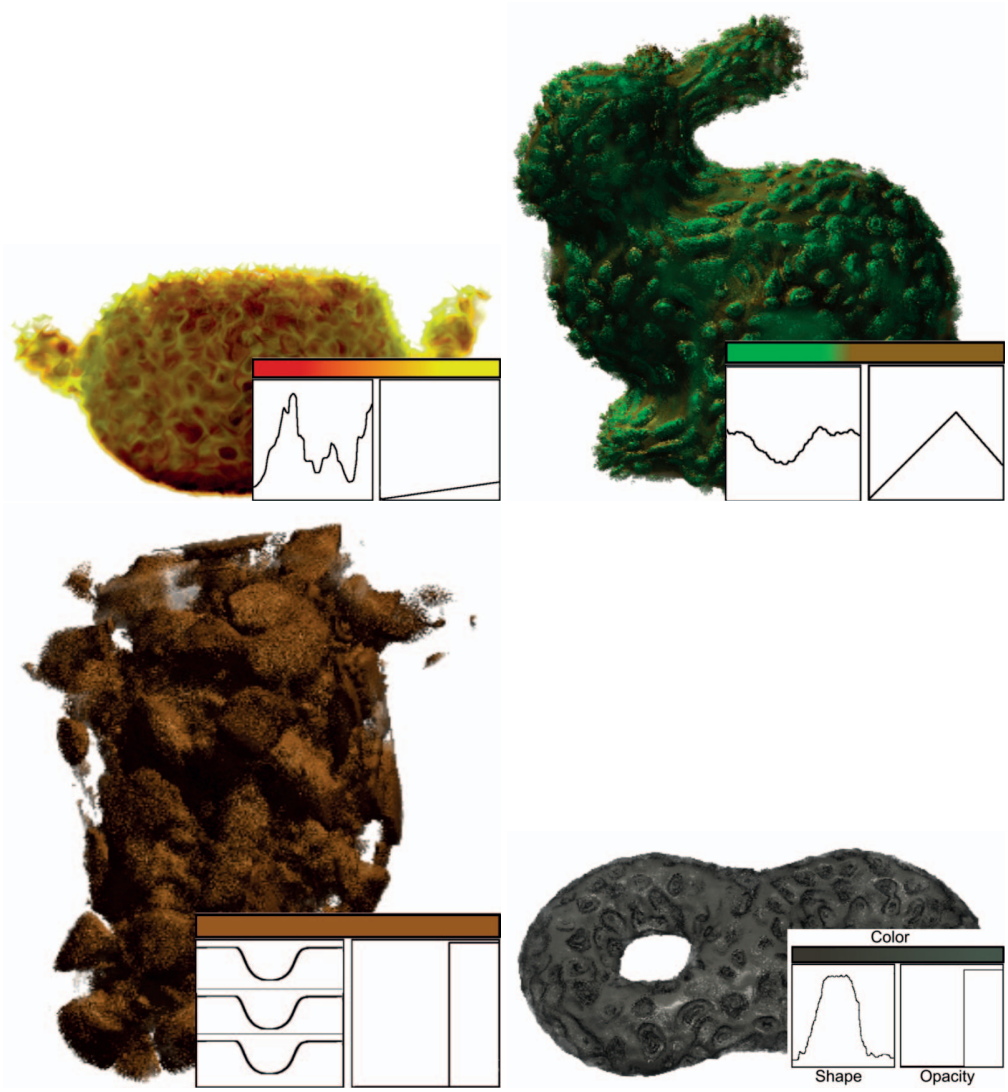


FIGURE 7.13 – Des exemples d'effets purement synthétiques sur des divers objets.

rendu plus sophistiqué. Ici, nous montrons un exemple des mottes de terre de la figure 7.12 appliquées à un champ et rendu avec un lancer de rayon de *Monte Carlo* en utilisant une *environment map* produite de manière synthétique et un système d'illumination globale (de type tracé de chemins). Cet exemple illustre également que grâce à la nature procédurale de notre modèle, il est possible de texturer des objets d'une taille arbitraire sans effet de répétition.

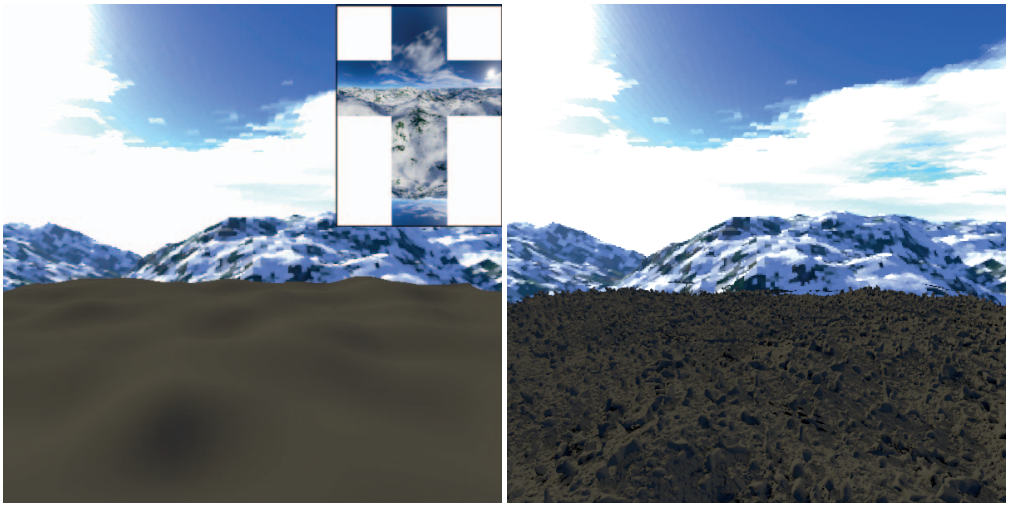


FIGURE 7.14 – Une fois que l'hypertexture a été définie, elle peut être utilisée dans un système de rendu plus sophistiqué (ici un algorithme de tracé de chemins avec de l'illumination globale) pour texturer des objets pouvant être de taille infinie.

7.6 Conclusion et discussion

Nous avons présenté dans ce chapitre une application de notre méthode d'ajout de détails de meso-structure. Cette application se concentre sur l'ajout de détails procéduraux au travers de l'édition et de la manipulation d'hypertextures. Notre approche propose notamment un contrôle accru du résultat final, même pour des utilisateurs ayant peu de connaissances en programmation.
Ces travaux sont basés sur une reformulation de la fonction de modulation de la densité, et plus précisement sur l'édition temps-réel des fonctions de transfert de forme et d'opacité. Ces fonctions, en conjonction avec notre méthode de rendu interactive permettant un retour visuel immédiat, offrent la possibilité à un utilisateur de contrôler l'apparence finale de l'hypertexture d'une manière intuitive et directe. De plus, le modèle d'hypertexture étant totalement dynamique, il devrait être possible d'obtenir des hypertextures animées, par exemple en rendant les fonctions de transferts dépendantes du temps, ou en utilisant une fonction de bruit 4D en place d'une fonction 3D, ou encore en changeant dynamiquement le champs de vecteurs.

Il est également possible d'améliorer ce modèle d'hypertexture. Premièrement, en ce qui concerne les fonctions de transfert de formes, il serait certainement extrêmement utile de proposer une méthode permettant de les construire automatiquement en utilisant un exemple. Il serait intéressant d'imaginer un système où l'utilisateur défini un échantillon de meso-structure avec un ou-

til 3D standard et le donne au système qui, en retour, analyse sa géométrie et construit de manière automatique une fonction de transfert de forme adéquate créant un effet visuel similaire. Autrement dit, cela consisterait à adapter des outils d'analyse et de synthèse de textures aux cas des meso-structures volumiques et aux déformations procédurales. Toutefois, cela semble de nos jours être un problème non-trivial et difficile.

En ce qui concerne notre méthode de rendu, cet exemple met en avant à nouveau la définition du contrôle de la qualité, et donc du taux d'échantillonnage. Nous pensons que, à l'instar du rendu volumique, ce paramètre pourrait bénéficier ici d'un schéma adaptatif, de manière à concentrer les calculs dans les parties utiles. Plus spécifiquement, dans cette application, nous pensons que ce paramètre pourrait être ajusté automatiquement, en prenant en compte par exemple la forme de la fonction de transfert de forme, ainsi que les paramètres d'amplitude et de fréquence. Toutefois, la nature procédurale et hautement stochastique des fonctions de bruit transforment l'introduction d'un tel schéma en un problème difficile.

Chapitre 8

Génération de textures semi-procédurales par l'analyse d'un exemple

Les travaux présentés dans ce chapitre rompent avec l'utilisation de notre méthode de rendu de mésostructure décrite précédemment et se concentrent sur la génération de textures en se basant sur l'analyse d'un exemple. Cette technique s'apparente à la notion de synthèse de textures. L'idée est ici de recréer lors du rendu l'apparence d'une surface en se basant sur une image d'exemple. C'est-à-dire qu'il s'agit de se placer dans l'espace paramétrique de la surface d'un objet et de générer des informations de couleur afin de lui donner une apparence « ressemblant » à celle de l'image d'entrée.

Cette méthode se base, à l'instar des hypertextures (voir chapitre 7), sur une définition procédurale basée sur une somme de bruits. Ces définitions, extrêmement compactes, permettent de texturer des environnements immenses avec un coût mémoire quasi-nul. Grâce au matériel graphique, il est possible de calculer en temps réel l'apparence d'une surface pour chaque fragment visible la composant. Cependant, la production d'une texture complexe ayant une apparence naturelle avec un simple bruit est un problème difficile. Il nécessite de solides connaissances de programmation et beaucoup d'essais avant d'obtenir le résultat désiré. Cette limitation est une des causes possibles du développement des méthodes de synthèse de textures basées sur l'analyse d'un exemple. Ces dernières laissent les utilisateurs fournir une image d'exemple, qui est analysée et reproduite automatiquement avec plus ou moins de contrôle sur l'apparence finale. En règle générale toutefois, ces méthodes sont radicalement différentes d'une méthode procédurale : la description de l'apparence n'est pas compacte car les textures doivent être pré-générées et stockées explicitement sur le matériel graphique. De plus, la taille et la résolution des textures ainsi générées restent limitées.
L'objectif de ces travaux est d'introduire une méthode qui, prenant en entrée une image, produit un ensemble compact de paramètres pour une description

semi-procédurale de la texture en utilisant une somme de fonctions de bruit (ainsi que des fonctions cosinus pour les parties périodiques). Cette technique se base sur la capacité à reproduire un signal aléatoire stationnaire [1] de manière fonctionnelle. Beaucoup de textures artificielles ou naturelles ont comme base un tel signal statique aléatoire, comme les textures purement aléatoires [60], les textures quasi-régulières [94] ou encore les textures basées sur une distribution aléatoire d'éléments visuels : tels les *texture particles* [40].

Nous commençons ce chapitre par quelques rappels sur l'analyse et la synthèse de texture (section 8.1). Nous continuons avec le principe de notre technique (section 8.3) avant de présenter une classification des textures dans la section 8.2. La section 8.4 contient les éléments concernant la synthèse procédurale d'un signal aléatoire et non structuré. Nous finissons par présenter la technique utilisée pour l'extraction des signaux aléatoires ainsi que les résultats correspondants (section 8.5).

8.1 Travaux antérieurs sur la synthèse de texture

La synthèse de texture à partir d'une image d'exemple est l'objet de nombreux travaux (nous renvoyons le lecteur vers le tour d'horizon proposé dans [159]), qui, pour la plupart, utilisent une image 2D en entrée et produisent en résultat une image 2D de taille arbitraire (certains toutefois produisent des textures solides [71]). Ces méthodes exploitent généralement l'autosimilarité des voisinages locaux et consistent en l'analyse de groupes de pixels pouvant être simplement un voisinage local [160, 4], des patches complets [93, 127, 45, 80] ou des éléments caractéristiques précédemment segmentés [40, 174]. Des travaux visant à la génération de textures en temps réel ont aussi été conduits, grâce à une implémentation efficace des algorithmes sur le matériel graphique [85, 86] ou à des structures de données précalculées [173].

En dépit de ce gain de vitesse, il est difficile de texturer une surface de taille arbitraire lors du rendu, principalement à cause de la nécessité d'obtenir des informations de voisinage. De ce fait, les textures sont majoritairement prégénérées soit sur un plan plaqué à postériori sur la surface soit directement sur la surface elle même [151, 161, 172]. Dans les deux cas, il est nécessaire de disposer de capacités de stockage importante, surtout dans le cas de textures à haute résolution ou de dimensions importantes, comme des textures 3D ou

1. On dit d'un signal qu'il est stationnaire si toutes ses propriétés statistiques sont invariantes dans le temps (ou l'espace).

même 3D animées.

Cela contraste fortement avec la nature même d'une méthode procédurale [44] où la texture peut être considérée comme définie de manière fonctionnelle pour chaque point d'un espace 2D ou 3D et permet de texturer lors du rendu des surfaces de taille arbitraire, voire infinie sans recourir à des données précalculées stockées en mémoire.

Afin de s'approcher des motifs naturels, qui peuvent souvent présenter un caractère aléatoire, les textures procédurales utilisent souvent des fonctions de bruit et des turbulences comme des outils pour « perturber » des fonctions de base régulière (par exemple, la fonction sinus utilisée par Perlin pour réaliser du marbre [120]). Beaucoup de travaux ont été conduits afin de définir des fonctions de bruit avec des propriétés visuelles, spectrales et statistiques très variées [120, 91, 92, 169, 122].

Ces méthodes procédurales requièrent toutefois des connaissances en programmation et une longue phase d'essais-erreurs avant d'obtenir le résultat souhaité. Afin de faciliter la définition de textures procédurales, Ghazanfarpour et al. proposent [54] de tirer d'une image d'« exemple » les paramètres pour créer des textures solides basées sur une perturbation. Cette image d'exemple est utilisée pour produire un fonction de bruit ayant des caractéristiques similaires (utilisant un filtre appliqué à un bruit blanc), ainsi qu'un jeu de fonctions régulières basées sur un ensemble de cosinus. Cela a été la première approche permettant d'obtenir une texture procédurale en analysant une image d'exemple. Cependant, la ressemblance avec l'image d'origine ne peut être garantie, car seule l'information de distribution spectrale est utilisée pour définir un champ de perturbations. Ces caractéristiques spectrales seules ne sont pas suffisantes pour capturer l'ensemble des caractéristiques visuelles d'une texture.

Une description procédurale de façades d'immeubles a été proposée récemment par Muller et al. [112]. Toutefois, cette technique ne se concentre pas sur un schéma de synthèse générique et utilise des connaissances à priori, limitant ainsi son application à d'autres types de surface.
Bourque et al. [20] proposent de rechercher les paramètres permettant la reproduction de l'apparence d'une texture d'entrée par un *fragment program* dans une base de données. L'idée est, partant d'une base de données de *fragment programs* paramétrés, de sélectionner un *fragment program* et de rechercher dans l'espace de ses paramètres la combinaison permettant de générer un résultat proche de l'image donnée en entrant. Cela implique la création d'un échantillon de texture et une comparaison avec le résultat final pour chaque

jeu de paramètres, ce qui peut causer en pratique une explosion combinatoire lorsque le nombre de paramètres est élevé. Par ailleurs, le résultat obtenu peut être médiocre et éloigné de l'exemple, selon qu'il existe ou non un *fragment program* adéquat dans la base de données.

Nos travaux sont inspirés de [39], où des courbes 1D (appelées « profils ») sont analysées et une description procédurale basée sur un bruit est créée en utilisant l'analyse combinée de l'histogramme et de la distribution spectrale de l'image. Ces deux analyses garantissent une ressemblance avec l'exemple fourni en termes de statistiques d'ordre un et deux. Ces profils ont été utilisés pour créer des cartes de déplacements, mais n'ont pas été appliqués à l'analyse de textures 2D.

8.2 Classification des textures

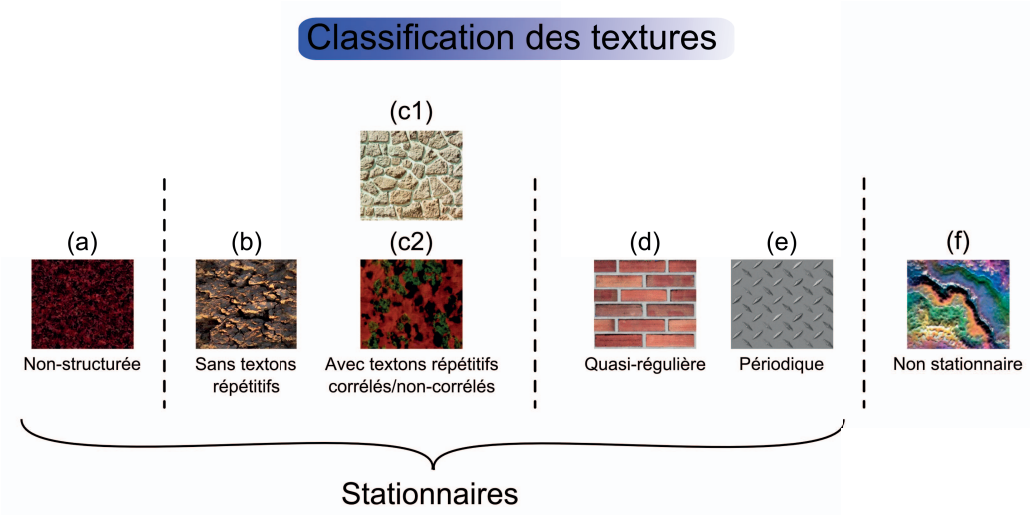


FIGURE 8.1 – Notre proposition de classification des textures.

Cette section propose une classification non-exhaustive des textures, illustrée par la figure 8.1. Cette classification s'inspire de celle proposée dans [94] et se propose de la compléter. Les textures peuvent être divisées en premier lieu par la nature stationnaire ou non de leur signal. La grande majorité des méthodes de synthèse de tetures se basant sur l'exploitation de l'auto-similarité d'une texture, la synthèse de textures non-stationnaires (f) reste un problème extrêmement difficile et peu abordé. Nous nous intéressons ici uniquement aux

textures ayant un signal stationnaire. Nous pouvons les diviser en plusieurs catégories, suivant la composition des signaux aléatoires sous-jacents. Nous pouvons en premier lieu distinguer les textures non-structurées (a), c'est-à-dire ne présentant aucun élément visuel caractéristique et composées exclusivement d'une fonction de bruit.
Pour les catégories restantes, les textures présentent des éléments visuels caractéristiques, appelés *textons* [63]. La répartition de ces textons permet de séparer les deux catégories restantes. Nous avons d'un côté une répartition de textons périodique ou quasi-périodique et de l'autre une répartition « aléatoire ». Pour le premier cas, nous pouvons distinguer les textures totalement périodiques (e), ou répartissant de manière « quasi-régulière » des textons d'apparence très similaires : les textures quasi-régulières (d).
Le second cas concerne les textures avec des textons distribués de manière non régulière. On peut distinguer dans cette catégorie les textures ne présentant pas de textons « répétitifs », c'est-à-dire composées d'un ensemble de textons « visuellement différents » (b), des textures contenant des textons répétitifs. Les textons « visuellement différents » représentent un élément de structure (comme par exemple les craquelures dans la figure (b), qui représentent une forme de structure) qu'il est difficile d'isoler et de mettre en correspondance avec un autre élément de la texture. Inversement, un mur de pierres, comme sur la figure ($c1$) est composé d'un ensemble de textons « visuellement similaires », c'est-à-dire représentant un même élément structurel de la texture mais soumis à des variations de forme ou de couleur. La répartition de ces textons peut être corrélée, c'est-à-dire que la « forme » et la répartition des textons dépend (totalement ou en partie) des textons voisins ($c1$) ou totalement décorrélée ($c2$).

Nous nous attachons dans ce chapitre à traiter les textures non structurées (a) et les textures présentant des textons répétitifs non corrélés (c2). Nous présentons toutefois dans la conclusion diverses pistes afin d'accroître la gamme de textures qu'il est possible de représenter procéduralement ou semi-procéduralement par notre méthode.

8.3 Principe

Le principe de notre approche est simple et se base sur le fait que beaucoup de textures (naturelles ou artificielles, voir la section 8.2) reposent sur un ou plusieurs signaux stationnaires aléatoires sous-jacents, c'est-à-dire que les signaux ont une certaine ressemblance avec du bruit ou de la turbulence. Ce phé-

nomène est largement exploité par la plupart des méthodes de génération de textures procédurales qui consistent à « perturber » un motif régulier ou périodique (comme pour le marbre de Perlin [120]). Il existe deux éléments permettant de caractériser un bruit : sa distribution de probabilités et sa distribution spectrale, c'est-à-dire sa distribution de l'énergie en fonction de sa fréquence.

La première étape de notre approche consiste à extraire ce signal d'une image d'exemple $I(i,j)$. Nous appelons $s(i,j)$ le signal résultant, qui est une image normalisée en niveau de gris à la même résolution que la texture d'exemple contenant un signal stationnaire aléatoire 2D. Un tel signal stationnaire est naturellement dépourvu d'éléments structurels mais il peut contenir des composantes directionnelles, voir périodiques.
La deuxième étape consiste à reproduire sous forme fonctionnelle le signal $s(i,j)$. Concrètement, nous proposons une méthode qui calcule automatiquement une fonction $s'(x,y)$ basée sur une somme de bruits et de cosinus telle que $ST(s') = ST(s)$, où ST représente les propriétés statistiques aux premiers ordres. Dans notre cas, ces statistiques sont de deux types : les statistiques de premier ordre (des histogrammes) et les statistiques de second ordre (une autocorrélation, qui peut être assimilée à une distribution spectrale d'énergie).
Finalement, la troisième étape consiste à dériver une texture procédurale $P(x,y)$ qui ressemble « visuellement » à la texture d'entrée $I(i,j)$ en utilisant la fonction aléatoire $s'(x,y)$. La texture résultante $P(x,y)$ est procédurale car elle est définie en fonction de l'espace (x,y) et peut être évaluée à la demande.
Les deux sections suivantes décrivent comment obtenir $s'(x,y)$ à partir de l'analyse du signal aléatoire stationnaire $s(i,j)$ (section 8.4) et comment synthétiser la texture procédurale $P(x,y)$ en procédant à l'extraction d'un ou plusieurs signaux aléatoires de l'image d'exemple (section 8.5).

8.4 Synthèse de signaux 2D aléatoires stationnaires

Dans cette section, nous décrivons notre approche pour créer une approximation fonctionnelle d'un signal stationnaire aléatoire $s(i,j)$ dont les valeurs sont normalisées entre 0 et 1. Nous considérons que deux signaux aléatoires sont « similaires » si les statistiques de premier et de second ordre sont identiques, c'est-à-dire si les histogrammes et les distributions spectracles d'énergie sont les mêmes. Notre but est d'utiliser des fonctions de bruit classiques $n(x,y)$ comme le bruit de Perlin [120, 122] (basé sur un gradient ou une méthode de simplexe). Nous définissons $s'(x,y)$ comme l'approximation fonctionnelle continue de $s(i,j)$ telle que $ST(s) = ST(s')$. Nous souhaitons exprimer $s'(x,y)$

comme une somme de bruits et de cosinus :

$$s'(x,y) = H\left[\sum_{i=0}^{n_n} A_i\, n(f_x^i x, f_y^i y) + \sum_{j=0}^{n_c} A_j \cos(2\pi(f_x^j x + f_y^j y))\right] \tag{8.1}$$

où H représente une fonction de transfert qui convertit les valeurs retournées afin de les faire correspondre avec un histogramme prédéfini. Les fonctions de bruit retranscrivent les composants aléatoires de $s(i,j)$ tandis que les cosinus représentent les composant réguliers.
La difficulté principale est de déterminer les paramètres H, n_n, A_i,f_x^i,f_y^i, n_c, A_j, f_x^j, f_y^j afin que $ST(s')$ soit aussi proche que possible de $ST(s)$, c'est-à-dire que $\left|ST(s') - ST(s)\right|$ soit minimal. Cette minimisation d'erreur pose toutefois un problème d'optimisation difficile qui ne peut être résolu directement. La solution consistant à balayer l'ensemble du champ de définition des paramètres et à évaluer le résultat [20] n'est pas viable ici en raison du nombre élevé de paramètres.
Pour simplifier le problème, nous proposons un ensemble d'heuristiques basées sur une étude expérimentale conduite sur les fonctions de bruit de Perlin. Cette étude est illustrée par les figures de 8.2 à 8.4. Nous proposons une technique en deux étapes qui estime dans un premier temps le nombre de bruits ainsi que leurs facteurs d'échelle, c'est-à-dire n_n, f_x^i and f_y^i et calcule ensuite les cosinus f_x^j, f_y^j, les amplitudes A_i et A_j et la fonction H. Ces deux procédures sont découplées, ce qui garantit une convergence vers une solution et la stabilité de la recherche de paramètres. Nous proposons ici de définir ces paramètres uniquement pour un bruit de Perlin et ne considèrons pas de ce fait toutes les formes de distributions spectrales.

Les sections suivantes décrivent plus en détail l'étude expérimentale, la méthode utilisée pour évaluer le bruit, ainsi que la technique calculant les paramètres des cosinus et des amplitudes. Finalement, la dernière section précise le calcul de la fonction de transfert H.

8.4.1 Étude expérimentale

Notre étude est basée sur le constat suivant : une fonction de bruit 2D classique (basée sur un gradient ou une méthode de simplexe) $n(f_x x, f_y y)$ mis à l'échelle avec deux facteurs f_x et f_y produit une forme elliptique dans le domaine fréquentiel, où les rayons de l'ellipse correspondant sont directement corrélés aux facteurs f_x et f_y. La figure 8.2 montre trois exemples pour le même bruit, avec trois couples de facteurs différents (f_x, f_y) respectivement égaux à $(1/32, 1/8)$, $(1/6, 1/6)$ et $(1/4, 1/8)$. La première ligne montre les trois images de

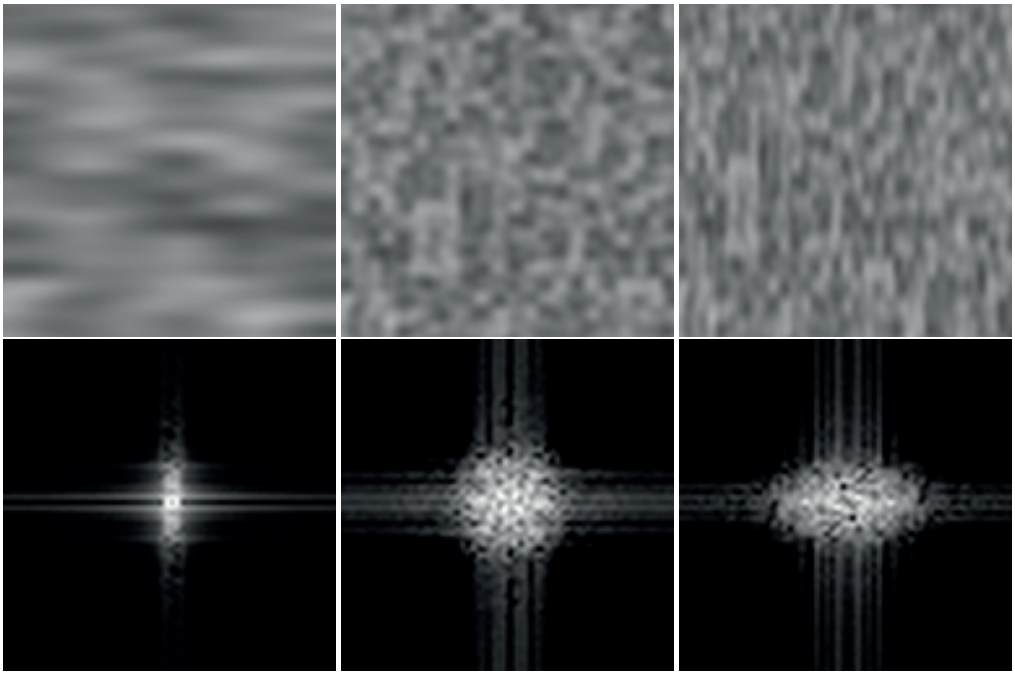


FIGURE 8.2 – Images de bruit classique avec trois facteurs d'échelle différents le long des axes x et y ainsi que leurs domaines fréquentiels respectifs.

bruit tandis que la ligne du bas illustre le domaine fréquentiel correspondant, calculé par transformée de Fourier rapide. On peut voir les formes elliptiques visibles dans le domaine fréquentiel, avec les rayons le long des axes x et y proportionnels à f_x et f_y. Il est à noter qu'il subsiste des fréquences parasites le long des axes, dues à l'interpolation polynomiale utilisée pour calculer le bruit.

L'idée est de reconstruire les facteurs d'échelle f_x et f_y par une technique d'extraction/segmentation des ellipses dans le domaine fréquentiel. La transformée de Fourier d'une fonction de bruit résulte également en du bruit, ce qui rend l'extraction automatique robuste des ellipses correspondantes difficile par une simple méthode de seuillage. Toutefois, l'intégrale du bruit est nulle sur l'ensemble du domaine. Cette propriété peut être exploitée afin d'améliorer la segmentation des ellipses.
Concrètement, nous pouvons calculer plusieurs jeux de transformée de Fourier pour des « coupes » de tailles variées, nous permettant ainsi de dériver une distribution fréquentielle d'énergie moyenne dans le domaine spectral. La présence de bruit dans le domaine fréquentiel est inversement proportionnel au nombre de coupes (plus le nombre de coupes est élevé, moins le bruit est présent dans le domaine fréquentiel). En effet, la distribution d'énergie moyenne

du bruit tend vers une valeur constante. Nos expériences ont montré qu'une segmentation est efficace même pour un nombre limité de coupes (environ 10).

Muni de ces informations, il est possible de déterminer l'ellipse qui corres-

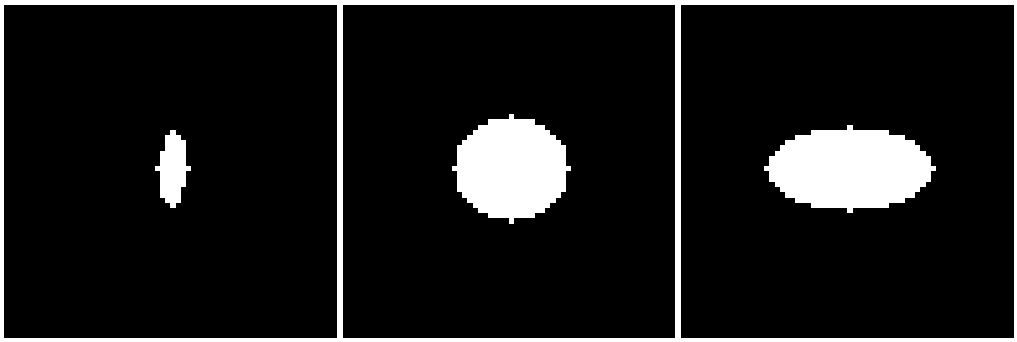


FIGURE 8.3 – Ellipses correspondant aux fonctions de bruit de la figure 8.2 recouvertes par technique de minimisation.

pond le « mieux » dans le domaine fréquentiel. Cette éllipse est déterminée par une procédure de minimisation. Nous voulons que l'ellipse soit calculée afin de maximiser le recouvrement du domaine fréquentiel segmenté. En prenant $F(i,j)$ l'ensemble des pixels du domaine fréquentiel binaire segmenté, et $E_{r_x,r_y,\alpha}(i,j)$ les pixels d'une ellipse de rayons r_x, r_y ayant subi une rotation d'angle α, alors le recouvrement maximum peut se définir par la formule de minimisation :

$$(r_x, r_y, \alpha) = min_{r_x,r_y,\alpha} \left| \left\{ E_{r_x,r_y,\alpha}(i,j) \oplus F(i,j) \right\} \right| \tag{8.2}$$

où $\oplus$ représente l'opérateur binaire « ou exclusif » et $||$ la cardinalité d'un jeu de pixels. La figure 8.3 montre les ellipses obtenues pour les trois bruits de la figure 8.2, avec des rayons respectifs de $(3,7)$, $(11,10)$ et $(16,8)$.

Cette procédure peut être utilisée pour déterminer toutes les ellipses correspondant aux divers facteurs d'échelle f_a (avec $a \in \{x,y\}$) des fonctions de bruit. Cela nous permet de précacluler une table de corrélation T associant à chaque rayon r_a un facteur d'échelle correspondant f_a. Étant donné qu'une rotation dans le domaine spatial implique une rotation du même angle dans le domaine fréquentiel, une seule table $f_a = T[r_a]$ valide pour les deux axes x et y peut être calculée, en interpolant les valeurs manquantes et moyennant les valeurs redondantes.

La figure 8.4 montre les deux tables obtenues pour les fonctions de bruit basées sur le gradient et sur le simplexe, en utilisant des coupes de 64×64 pixels.

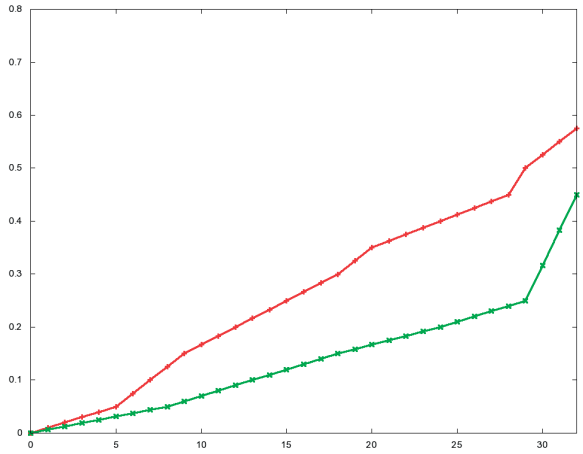


FIGURE 8.4 – Fonctions obtenues pour la fonction de bruit de Perlin basée sur le gradient (en rouge) et sur une techique de simplexe (en vert). Ces fonctions mettent en correspondance le facteur d'échelle et la fréquence.

En appliquant ces tables aux ellipses des figures 8.3, nous obtenons les facteurs d'échelle $(0.03, 0.1)$, $(0.18, 0.16)$ et $(0.26, 0.125)$, ce qui est très proche des facteurs d'échelle réels $(1/32, 1/8)$, $(1/6, 1/6)$ et $(1/4, 1/8)$.

8.4.2 Couverture du domaine fréquentiel

Nous exploitons ici le fait qu'une fonction de bruit produit des structures elliptiques dans le domaine fréquentiel pour déterminer les paramètres n_n, f_x^i et f_y^i définissant $s'(x, y)$. Premièrement, plusieurs coupes de $s(i, j)$ sont utilisées pour calculer le domaine fréquentiel moyen, comme illustré sur la figure 8.5. En haut de cette figure, on peut voir un signal $s(i, j)$ (à gauche) ainsi que le domaine fréquentiel moyen correspondant (au milieu). Nous procédons en quantifiant le domaine fréquentiel en q valeurs. Cette variable q est définie par l'utilisateur et a une influence sur le nombre de bruits n_n générés. Cette quantification est basée sur une échelle logarithmique afin de mieux respecter la décroissance exponentielle de l'énergie dans le domaine fréquentiel lorsque les fréquences augmentent.

L'image en haut à droite de la figure 8.5 montre le domaine quantifié pour $q = 4$. Pour chaque valeur $v = 0, 1, \ldots, q-1$, une image binaire peut être calculée, comme montré sur la seconde ligne de cette figure. Les images binaires sont obtenues en sélectionnant les pixels dans le domaine fréquentiel dont la valeur quantifiée est supérieure ou égale à v. Pour chaque image binaire, nous déterminons un ensemble minimal d'ellipses centrées sur l'origine qui recouvrent le mieux cette image. Il est nécessaire de définir les paramètres correspondant

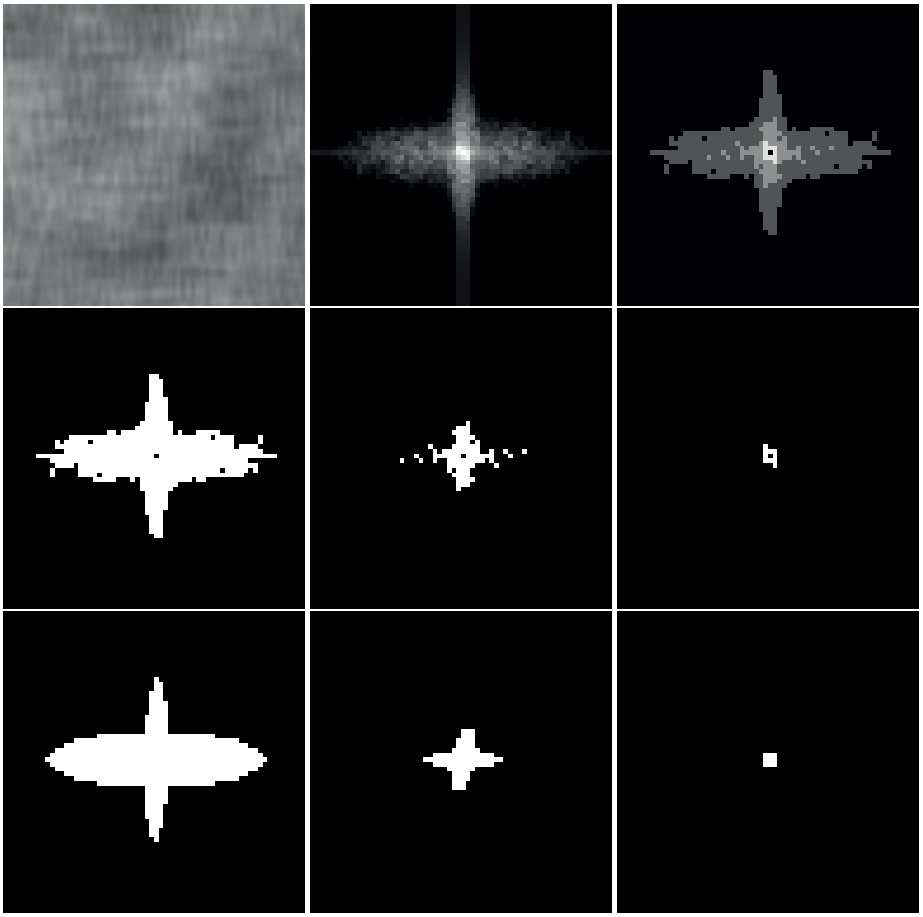


FIGURE 8.5 – Analyse d'un signal d'entrée $s(i,j)$ dans le domaine fréquentiel moyen par quantification en calculant les ellipses maximisant la couverture des pixels.

à l'angle de rotation de l'ellipse ainsi que ses deux rayons. Nous commençons par une seule ellipse maximisant le recouvrement en utilisant la formule de minimisation définie précédemment. Des ellipses supplémentaires sont ajoutées par un processus itératif jusqu'à ce que la couverture ne puisse plus être améliorée. Nous prenons pour contrainte que chaque ellipse doit couvrir un certain pourcentage de pixels pour être créée. Cela nous permet d'éviter la création d'ellipses en recouvrant d'autres et ne contribuant pas de manière significative au résultat final. La troisième ligne de la figure 8.5 montre le résultat obtenu pour les trois images binaires de la ligne précédente. Le nombre global d'ellipses est le paramètre n_n recherché ($n_n = 5$ sur la figure). Les facteurs d'échelle f_x^i and f_y^i sont obtenus via les rayons des ellipses et la table T.

8.4.3 Amplitude et composants réguliers

Dans cette section, nous discutons de la méthode utilisée pour déterminer les paramètres restants de $s'(x,y)$. Les paramètres pour les cosinus n_c, f_x^j, f_y^j sont également calculés dans le domaine fréquentiel moyen et sont isolés grâce aux maxima locaux d'énergie.
Il est toutefois plus difficile de déterminer les amplitudes A_i et A_j, car lorsque deux bruits avec différents facteurs d'échelle et différents poids sont additionnés, le domaine fréquentiel moyen n'est pas la somme pondérée des domaines fréquentiels moyens individuels. De plus, la fonction de transfert H a aussi une influence sur la distribution d'énergie dans le domaine fréquentiel.

Nous nous basons sur une technique itérative afin d'ajuster progressivement les amplitudes A_i et A_j. Nous assignons en premier lieu des valeurs initiales proportionnelles à l'énergie recouverte dans le domaine fréquentiel pour A_i et A_j (en se basant sur les images binaires obtenues au cours de l'étape précédente). Ensuite, l'histogramme de la fonction de transfert H est calculé (voir section 8.4.4). Cela nous fournit une première approximation des valeurs de $s'(x,y)$. Plus particulièrement, il est possible d'extraire des coupes de $s'(,y)$ afin de calculer le domaine fréquentiel moyen.
Ce domaine fréquentiel moyen peut être comparé à celui du signal d'entrée $s(i,j)$. Afin de faire correspondre les deux domaines, les rapports d'énergie peuvent être calculés afin d'ajuster les valeurs d'amplitude A_i et A_j. Si l'énergie du domaine fréquentiel moyen de $s'(x,y)$ est plus faible (resp. plus fort) que l'énergie correspondante de $s(i,j)$, alors l'amplitude correspondante est légèrement augmentée (resp. diminuée). Une nouvelle fonction de transfert H peut ainsi être calculée, permettant le calcul d'un nouveau signal $s'(x,y)$. Ce processus complet est réitéré afin d'obtenir des valeurs d'amplitude ajustées (le nombre d'itérations est défini par l'utilisateur).
La figure 8.6 illustre le résultat final obtenu pour le signal $s(i,j)$ de la figure 8.5.

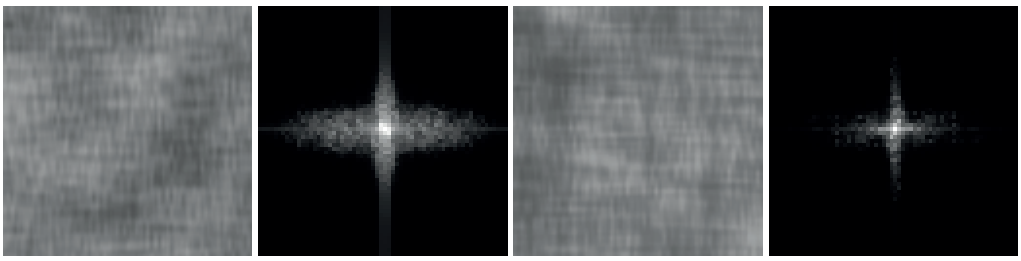


FIGURE 8.6 – La reproduction d'un signal aléatoire stationnaire défini comme une somme de cinq bruits (à gauche) à partir du signal d'entrée $s(i,j)$ (à droite).

La partie gauche représente le signal d'entrée $s(i,j)$ et son domaine fréquen-

tiel moyen. À droite, on peut voir le résultat $s'(x,y)$ défini avec une somme de cinq bruits (sans cosinus) et son domaine fréquentiel correspondant. Comme on peut le constater, $s(i,j)$ et $s'(i,j)$ sont similaires. Ce signal d'entrée $s(i,j)$ a été généré artificiellement grâce à une somme de trois fonctions de bruit (basées sur un gradient). Bien que deux bruits supplémentaires aient été extraits, notre méthode produit des résultats proches du signal d'entrée.

8.4.4 La fonction de transfert H

Le principe du calcul de l'histogramme H est simple et consiste en une technique d'égalisation d'histogramme classique souvent utilisée par les applications de traitement d'images. Prenons s_1 et s_2 deux images en niveau de gris de résolution similaire $R_x \times R_y$ contenants des entiers entre 0 et N. Les histogrammes $H_1[0..N]$ et $H_2[0..N]$ peuvent être calculés pour s_1 et s_2, définissant pour chaque valeur entière $i \in [0,N]$ sa fréquence, c'est-à-dire le nombre d'occurences de la valeur i.
Nous avons $\sum_{i\in[0,N]} H_1 = \sum_{i\in[0,N]} H_2 = R_x \times R_y$. Il est possible de définir une table de transfert H qui convertit les valeurs de s_1 afin que l'histogramme résultant soit « proche » de celui de s_2, c'est-à-dire que $H_1(H(s_1)) \approx H_2(s_2)$. Il est à noter qu'une correspondance exacte n'est pas toujours possible en raison de la nature discrète de s_1 et s_2.

Cette table de transfert d'histogrammes H est calculée en utilisant une méthode récursive binaire. En premier lieu, les indices médians de H_1 et H_2 sont calculés : les plus petits indices respectifs i_1 et i_2 tels que $\sum_{i\in[0,i_1]} H_1$ et $\sum_{i\in[0,i_2]} H_2$ soient supérieurs ou égaux à $(R_xR_Y)/2$. La création de la table H consiste à convertir toutes les valeurs de s_1 de 0 à i_1 en valeurs de s_2 entre 0 to i_2. Une fois que i_1 et i_2 ont été calculés, ce procédé est récursivement réitéré en considérant les deux sous-tables $H_1[0..i_1]$ et $H_1[i_1+1..N]$ jusqu'à atteindre une table ne contenant qu'un seul élément. En ce qui concerne l'application de cette technique à la définition $s'(x,y)$, la table de transfert d'histogramme H est obtenue en appliquant la procédure décrite précdemment en quantifiant les valeurs $s'(x,y)$.

8.5 Extraire les signaux aléatoires

La section précédente concernait la reproduction d'un signal aléatoire stationnaire. Celui-ci ne définit pas en lui-même une texture, mais peut être utilisé comme élément de base à la synthèse d'une texture procédurale ou semi-procédurale. Nous discutons dans cette section de l'extraction des signaux aléa-

toires sous-jacents de l'image d'exemple afin de produire la définition procédurale. Nous nous concentrons spécifiquement sur deux types de textures pour lesquelles l'extraction du bruit sous-jacent est triviale : les textures totalement aléatoires sans composants structurels et les textures à particules caractérisées par une répartition plus ou moins aléatoire de caractéristiques visuelles (voir section 8.2). Ces deux types de textures ont été le sujet de nombreux travaux mais aucune définition procédurale n'a été proposée.

8.5.1 Textures non-structurées

Le premier type de texture peut être directement traité par notre approche décrite dans la section précédente en considérant la texture en elle-même comme un signal aléatoire stationnaire. Le seul problème posé est celui de la couleur (dans la section précédente, nous considérions uniquement des images en niveau de gris). La couleur peut être traitée en utilisant une décomposition en composantes principales afin de décorréler les trois canaux de couleur RVB [60]. L'avantage d'une telle décorrélation provient de la linéarité de la transformation de l'espace RVB en ce nouvel espace de couleur. Cette transformation peut donc être calculée avec un simple produit matriciel, ce qui présente l'avantage de pouvoir être réalisé efficacement au sein d'un *fragment program*. Nous obtenons pour chaque canal de couleur un jeu de paramètres définissant des bruits et des cosinus, une matrice 4×4 pour la transformation d'espace de couleur ainsi que trois tables de transfert d'histogrammes. Cela nous fournit une description extrêmement compacte et continue de la texture (en général moins de 1Kb). Nous utilisons l'implémentation matérielle du bruit de Perlin proposée par Gustavson [58].

Il est de plus aisément possible d'augmenter la portée de la méthode en étendant cette génération à l'espace 3D via l'utilisation d'un bruit 3D, la troisième fréquence pouvant être choisie de manière arbitraire (égale à la fréquence en x ou y par exemple). De plus, l'animation devient possible naturellement en introduisant une fonction de bruit variant en fonction du temps.

Nous avons conduits plusieurs tests [2] avec diverses images d'entrée. La figure 8.7 montre quatre exemples de textures aléatoires non-structurées (de haut en bas, du granit, de la peau, du bois et de la rouille). Pour chacune, nous obtenons une définition fonctionnelle très compacte, interprétée par un *fragment program* décrit par l'algorithme 8.1. Ce fragment program prend notamment

2. Notre environnement de test consiste en un Intel Core 2 Duo et une GeForce 8800GTX.

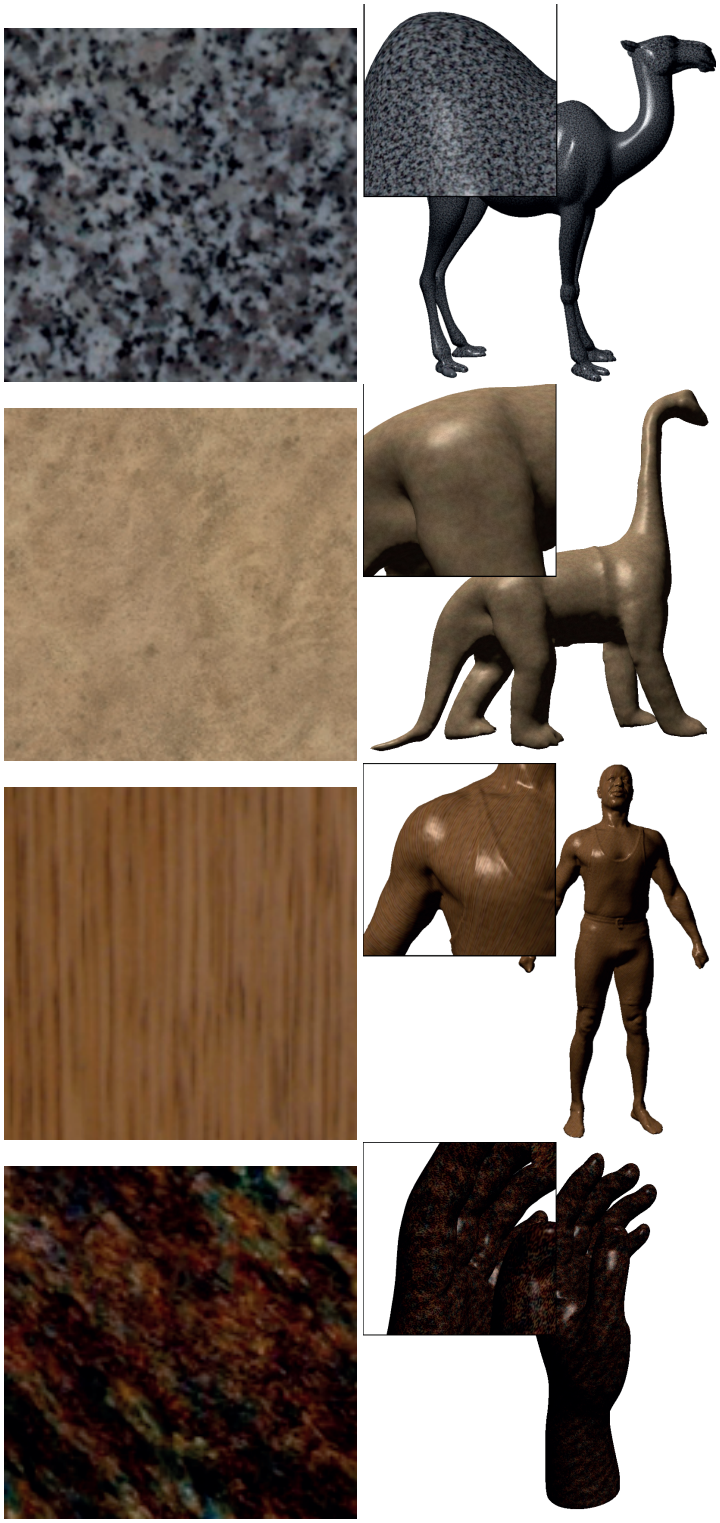


FIGURE 8.7 – À gauche : l'image d'exemple. À droite : un rendu (> 60 fps) d'une texture procédurale similaire sur un modèle 3D en se basant sur une somme de bruits et de cosinus.

en entrée un paramètre de fréquence modifiant les coordonneés de textures. Ceci permet de modifier l'échelle de l'espace 2D paramétrique à la surface de l'objet et influe sur l'échelle de la texture générée.

Algorithme 8.1 Implémentation directe de la méthode proposée pour générer une texture basée sur un bruit stationnaire.

ENTRÉES: Transfo : La matrice 4×4 de transformation d'espace de couleur
ENTRÉES: H_i : Fonction 1D de transfert d'histogrammes pour chaque composante de couleur
ENTRÉES: N_i : Un ensemble de fonctions de bruits paramétrés pour chaque composante de couleur
ENTRÉES: C_i : Un ensemble de fonctions cosinus paramétrées pour chaque composante de couleur
ENTRÉES: texcoord : Des coordonnées de texture pour chaque fragment
ENTRÉES: Freq : La fréquence globale de la texture
SORTIES: color : La couleur finale

```
Rasterisation d'un modèle surfacique paramétré
pour tout fragment f faire
  texcoord ← Freq ∗ texcoord
  pour tout i ∈ R,V,B faire
    pour tout n ∈ N_i faire
      color_i ← color_i + n()
    fin pour
    pour tout c ∈ C_i faire
      color_i ← color_i + c()
    fin pour
    color_i ← H_i(color_i)
  fin pour
  color ← Transfo ∗ color
  return color
fin pour
```

Notre approche peut directement être étendue pour produire des textures solides en se basant sur un bruit 3D (voir fig 8.8). De plus, cette texture peut être animée en utilisant un bruit de dimension supérieure.

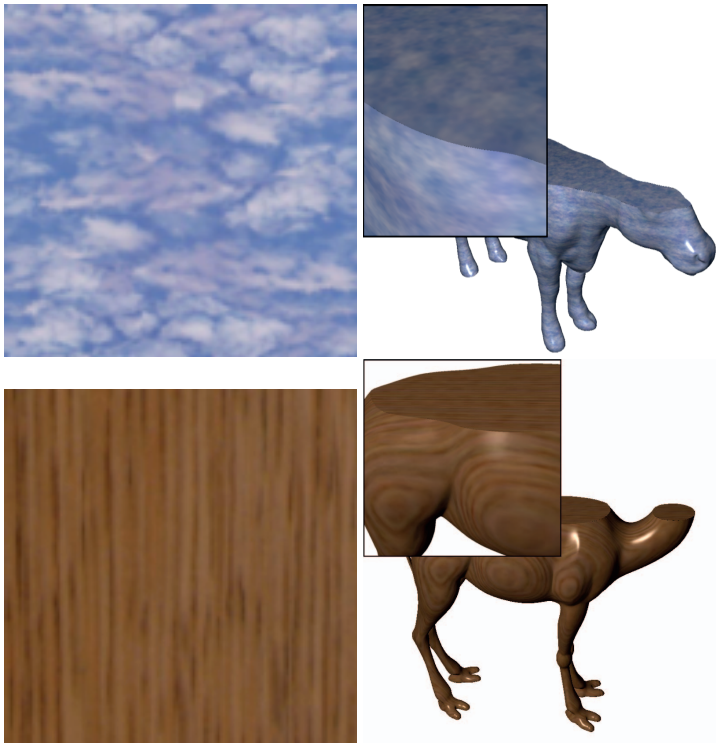


FIGURE 8.8 – Extension de notre méthode à l'application de textures solides et animées en utilisant une texture 2D d'entrée.

8.5.2 Textures à « particules »

La deuxième catégorie de textures, les textures à particules sont traitées en utilisant une approche similaire à [40]. Le principe est d'extraire de la texture d'entrée un ensemble de particules caractéristiques et de générer une nouvelle texture en replaçant ces éléments sur une image de fond. Dans [40], des couches de particules sont extraites et traitées séparement. Pour chaque couche, un jeu de propriétés caractéristiques est extrait et une image de fond est générée.

Contrairement à [40], nous proposons l'utilisation d'un signal aléatoire stationnaire pour déterminer l'emplacement des éléments extraits, ce qui nous permet d'obtenir une définition semi-procédurale de la texture (semi-procédurale, car nous utilisons toujours des textures 2D pour représenter les particules). Pour cela, nous prenons comme hypothèse que la position de ces particules, initialement régulière et périodique, est perturbée par une fonction aléatoire stationnaire. Cela présente des similarités avec les textures « quasi-régulières » [94]. Afin d'obtenir les positions des particules, nous laissons le soin à l'utilisateur de dessiner un rectangle couvrant un ensemble de particules formant un des motifs réguliers de la texture d'entrée. À partir de ce rectangle, nous pouvons extraire un sous-ensemble de particules ayant leur barycentre dans ce rectangle. C'est le jeu de particules qui sera repété et perturbé. Afin de déterminer les paramètres de la perturbation, le rectangle est déplacé sur la gauche, la droite, le haut et le bas de sa hauteur et largeur. Pour chaque particule déplacée contenue dans le rectangle, la position de la particule la plus proche est recherchée dans l'image exemple. Pour chaque particule, il est donc possible de calculer une mesure d'un vecteur de « perturbation ». Ce vecteur est quasiment nul dans le cas de textures périodiques. Deux signaux aléatoires $s_x(i, j)$ et $s_y(i, j)$ sont ainsi obtenus dans le rectangle par interpolation des coordonnées du vecteur par une convolution avec un noyau gaussien.

La couleur d'un point (x, y) est déterminée simplement en cherchant la particule la plus proche. Afin que les particules puissent se recouvrir, il est nécessaire de considérer les deuxièmes et troisièmes particules les plus proches. Cette recherche est la partie la plus coûteuse de notre méthode. Toutefois, nous montrons dans la section suivante une implémentation matérielle de cette recherche. La texture étant définie de manière fonctionnelle pour chaque point de l'espace 2D, il est également possible de paramétrer ces particules, par exemple en modulant aléatoirement la taille des particules. Notons que Lefevbre et al. proposent [87] un principe similaire de textures composites pour la génération procédurale de textures basées sur des particules mais ils n'utilisent pas

d'exemples afin de reproduire une distribution spatiale particulière des particules.

Afin de réaliser cette méthode sur le GPU, il est nécessaire d'obtenir pour chaque pixel synthétisé l'emplacement de la particule la plus proche. Itérer sur l'ensemble des particules afin de trouver la plus proche peut dégrader sévèrement les performances, même dans un *fragment program*. Afin de conserver des performances interactives, nous proposons une technique basée sur une méthode de splatting. Les particules sont représentées par les primitives points OpenGL et projetées lors du rendu, en utilisant la distribution spatiale associée, dans un tampon intermédiaire, que nous appelons *bomb map*. Pour chaque pixel recouvert par une particule, nous écrivons dans le tampon la position de la particule dans l'espace texture, en utilisant la précision complète à 16 bits du matériel graphique.
Lors du rendu, la lecture de cette carte donne pour chaque pixel la position de la particule la plus proche. La distance exacte entre le pixel et la particule est calculée en utilisant toute la précision du *fragment program* et permet de dériver des coordonnées de texture. Ces coordonnées de texture précises seront utilisées pour indexer la texture de particule. Cela nous permet de représenter chaque particule avec la résolution de la texture d'origine indépendamment de la taille de la surface texturée tout en maintenant des performances temps réel. Dans le cas où plusieurs particules s'intersectent, nous avons à utiliser plusieurs *bomb map*, donnant ainsi non seulement la particule la plus proche mais la seconde ou la troisième. En pratique, nous limitons toutefois le nombre de particules s'intersectant à deux par couches.

La figure 8.9 illustre une texture basée sur un ensemble de particules. Alors que la colonne de gauche montre l'image d'entrée, celle de droite illustre le résultat de notre méthode. Cette description semi-procédurale nous permet de reproduire des textures « infinies similaires » à la texture d'origine avec des performances temps réel. De plus, la densité des particules, ainsi que leur taille, peut être éditée au vol, permettant ainsi de modifier l'apparence visuelle de la texture tout en gardant une forte ressemblance avec la texture d'origine, comme illustré par la figure 8.10. Le fragment program 8.2 présente le *fragment program* utilisé lors du rendu (pour une seule couche de particule). C'est une implémentation directe de la méthode présentée précédemment.

Comme pour les « textures particles », notre méthode peut être utilisée pour ajouter de nouvelles particules (extraites d'autres images, par exemple) sur n'importe quelle texture avec la résolution de l'image d'origine (voir figure 8.11).

Finalement, la figure 8.12 montre une comparaison de notre méthode avec une méthode de *quilting* [45] (au milieu). Comme le montre cette figure, nos ré-

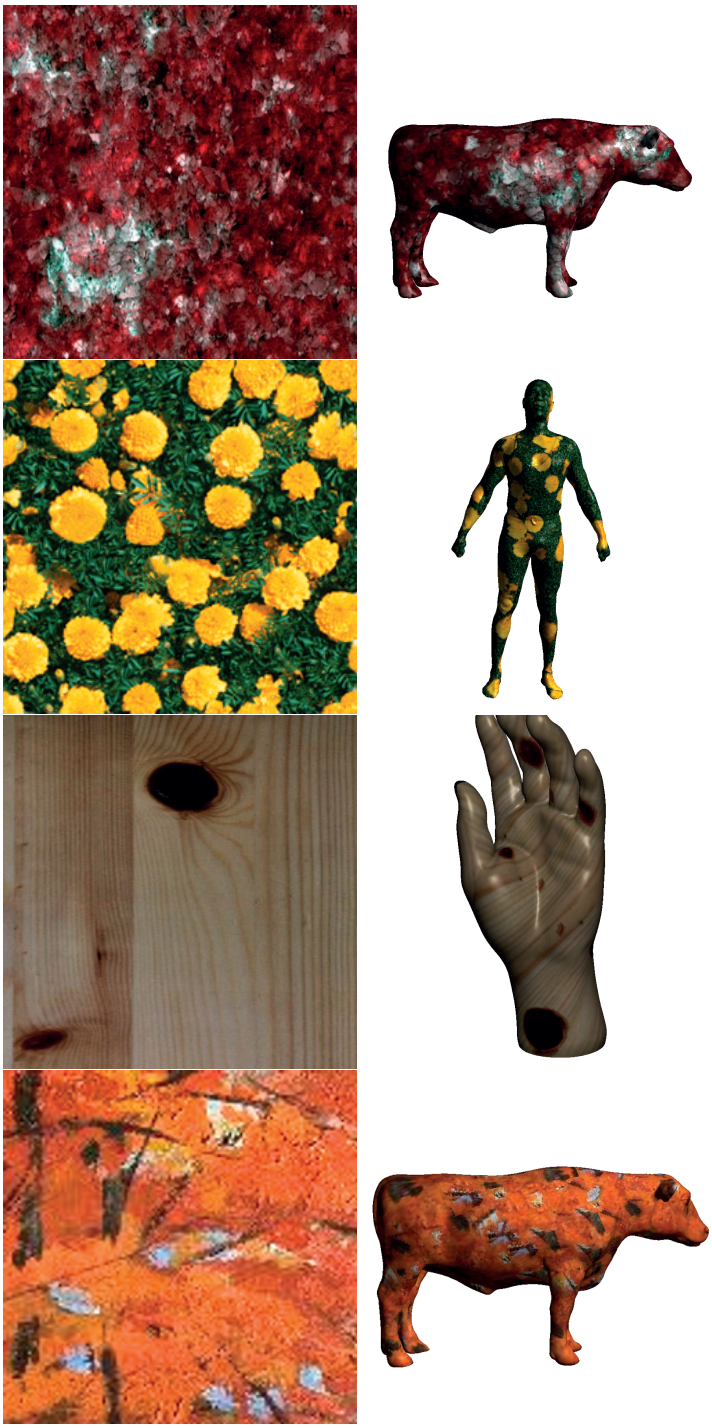


FIGURE 8.9 – Reconstruction au vol de textures en utilisant une distribution procédurale des particules (> 60 fps).

Algorithme 8.2 Pseudo-code utilisé pour notre méthode de *bombing*.

Première passe

ENTRÉES: P : un ensemble de particules
ENTRÉES: bombmap : un tampon RVBA intermédiaire
pour tout $p \in P$ **faire**
Perturbation de la position (2D) de la particule
Rasterisation du carré englobant de la particule
Ecriture pour chaque fragment des paramètres de la particule (position exacte du centre, type et orientation)
fin pour

Deuxième passe

ENTRÉES: TexParticule : la texture de la particule
ENTRÉES: bombmap : le tampon précédemment généré (utilisé ici comme une texture)
Rasterisation d'un modèle surfacique paramétré
pour tout fragment f **faire**
Génération d'une couleur de fond (par une texture ou une méthode procédurale)
Détermination d'une particule en indexant la texture de bombmap
Lecture de la bombmap aux coordonnées de texture du fragment
si Présence d'une particule **alors**
Calcul des coordonnées de textures relatives à la particule
Lecture d'une couleur en indexant la texture de la particule
Mélange de la contribution avec le fond
finsi
return Contribution finale du fragment
fin pour

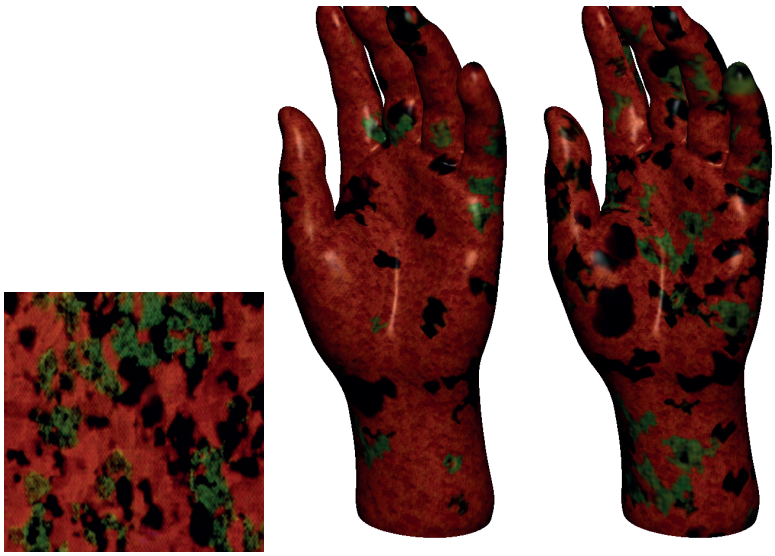


FIGURE 8.10 – Les paramètres, comme la densité des particules peuvent être édités au vol (>60 fps).

sultats (sur la droite) sont compétitifs avec le *quilting*, bien que notre méthode repose sur une description procédurale. En fait, nos résultats sont équivalents aux *textures particles*[40] (qui proposent une comparaison exhaustive avec les méthodes existantes) sauf qu'ici, le positionnement des particules est procéduralement généré en temps réel lors du rendu.

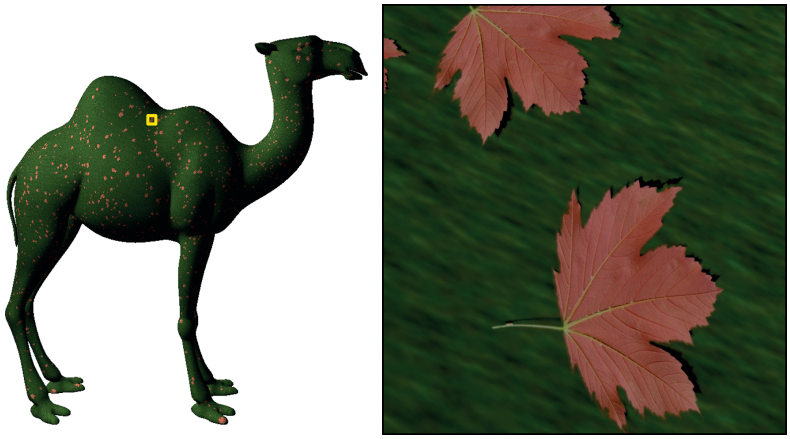


FIGURE 8.11 – À gauche : vue de loin d'une texture très haute résolution. À droite : Zoom sur une particule illustrant le fait que notre texture semi-procédurale préserve l'ensemble des détails visuels et n'a pas été sous-échantillonnée.

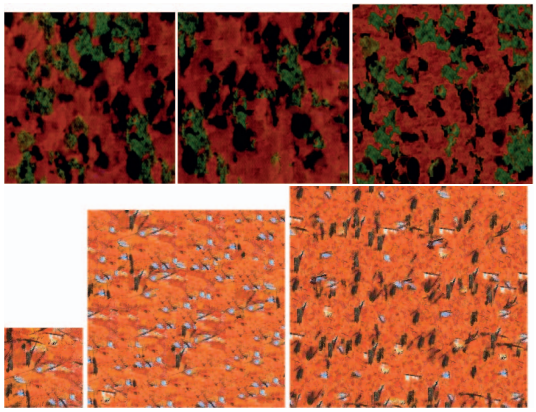


FIGURE 8.12 – Comparaison entre notre méthode (à droite) et une méthode de *quilting* (à gauche). Nos résultats sont équivalents à [40] mais générés en temps réel lors du rendu.

8.6 Conclusion

Nous avons présenté une approche de synthèse de textures basée sur l'analyse d'exemple. Contrairement aux techniques précédentes, notre approche est basée sur une desription procédurale des textures, ce qui a de nombreux avantages : les textures sont fonctionnellement définies sur un espace 2D ou 3D infini, des textures à très hautes résolutions peuvent être créées sans surcoût mémoire, la synthèse est effectuée à la demande, ce qui nous autorise à texturer des environnements de très grande taille. De plus, l'implémentation matérielle nous offre des performances temps réel tandis que la paramétrisation possible de nos modèles nous autorise à modifier au vol les aspects visuels des textures,

comme la densité ou la taille des particules.
Toutefois, il reste encore quelques inconvénients. Le principal est la gamme limitée de signaux stationnaires qu'il est possible de représenter. En effet, nous avons pris pour hypothèse que le domaine fréquentiel pouvait être partitionné en utilisant des ellipses centrées sur l'origine, ce qui n'est pas toujours le cas pour certains signaux, comme les distributions de Poisson. Le second point provient du fait que nous avons limité les textures structurées aux particules de texture. Cela ne permet pas de traiter d'autres types de textures, comme les mosaïques par exemple, ou lorsque les positions des particules sont corrélées. Toutefois, de telles textures structurées reposent également sur un signal aléatoire sous-jaçent perturbant une grille régulière. Une des extensions futures de ces travaux consisterait à extraire les paramètres de cette perturbation de grille régulière afin de permettre la génération de textures quasi-régulières en utilisant notre technique de « bombing ». Précisement, une particule serait représentée non plus par un carré matérialisé par un point OpenGL mais par un ensemble de polygones, ce qui permettrait d'une part de déformer la particule de manière plus précise et d'autre part de faciliter la corrélation des positions des particules. Finalement, nous souhaiterions étendre notre approche afin de traiter des distributions d'énergie plus complexes en utilisant d'autres fonctions de bruit aux caractéristiques spectrales différentes, comme par exemple le bruit de Gabor proposé dans [82].

Chapitre 9

Conclusion

9.1 Bilan

Nous avons présenté tout au long de cette thèse diverses solutions pour répondre au besoin croissant de richesse visuelle nécessaire pour la visualisation réaliste dans le domaine de la synthèse d'image. Précisement, nous avons proposés diverses méthodes se focalisant sur la modélisation et la représentation de détails de mésostructure afin d'enrichir l'apparence d'objets arbitraires. Plusieurs points ont été prédominants dans nos travaux :

- l'**interactivité** des méthodes de création et de rendu, en veillant à concevoir une méthode de rendu offrant un **contrôle** efficace sur le ratio qualité/performance des applications finales.
- la **généricité** de nos méthodes, en s'attachant à définir des méthodes permettant le traitement de la plus large gamme possible de mésostructures.
- la **souplesse de création**, notamment en proposant une **automatisation** des procédés de création offrant un **contrôle** du résultat visuel final.

Nos contributions sont résumées dans les sections suivantes. Nous terminons ce chapitre par quelques perspectives plus générales concernant ces travaux.

9.1.1 Rendu de mésostructure par lancer de rayon interactif

> **Idée** : Accroître la richesse visuelle d'un modèle (volumique ou surfacique) en effectuant le rendu de mésostructures volumiques par *fragment* en utilisant une méthode de *ray marching* permettant un contrôle sur la qualité et les performances.

Nous avons présenté dans le chapitre 4 une méthode de rendu hybride permettant l'ajout de détails de mésostructures volumiques à un modèle(surfacique ou volumique). Cette méthode hybride se base sur l'utilisation conjointe des principes de rastérisation et de lancer de rayons : la macrostructure du modèle est rendue par rasterisation tandis que le rendu de la mésostructure s'effectue pour chaque fragment par une méthode de *ray marching*.
L'un des intérêts principaux d'une méthode basée sur un calcul de mésostructure par fragment est d'associer le coût de la richesse visuelle d'un modèle à sa taille à l'écran. De plus, notre technique peut s'intégrer de manière naturelle dans les moteurs classiques existants et reste suffisamment générique pour s'appliquer à une large gamme de mésostructures. L'un des autres intêret majeur de notre méthode réside également dans sa facilité à utiliser un mécanisme de niveau de détails permettant de simplifier les calculs pour un objet peu important visuellement. Toutefois, il est à noter que les performances de notre méthode, du fait de sa généricité, présente des performances parfois moindres par rapport à des méthodes spécifiques à un type plus simple de mésostructure.
Pour résumer, voici les principaux avantages de notre méthode :

- Un coût de rendu en fonction de l'importance visuelle
- Une généricité permettant une large gamme de mésostructure
- Une approche surfacique et volumique
- Un schéma de niveau de détail

On peut toutefois noter quelques limitations :

- Des performances moindres que des méthodes ad hoc
- Une mésostructure contenue à l'intérieur du modèle

La perspective majeure pour l'amélioration de notre méthode consiste principalement en l'application d'un schéma d'échantillonnage adaptatif pour la partie de *ray marching*. Cette technique permettrait de concentrer les calculs principalement dans les zones très détaillées afin de permettre une amélioration des performances sans perte de qualité.

9.1.2 Un modèle à deux niveaux pour le rendu volumique

> **Idée** : Augmenter les performances des méthodes de rendu par splatting en utilisant notre méthode de rendu hybride sur un modèle volumique décomposé en deux niveaux structurels : un nuage de splats de tailles variables et un ensemble de textures 3D.

Ce chapitre 6 a présenté une première application de la méthode de rendu hybride en illustrant le rendu de mésostructure définies de manières explicite par une texture 3D. Ce rendu est permis en découplant un modèle volumique en deux parties. Chacune de ces parties correspond à un niveau structurel différent : une macrostructure représentée ici par un ensemble splats de taille variable et une mésostructure, définie par un ensemble de textures 3D. Un des points intéressant de cette méthode est une gestion du phénomène d'aliassage grâce à sa représentation à base de splats. Par ailleurs, notre méthode permet d'augmenter les performances des méthodes à base de splatting. Bien que ces performances soient inférieures au techniques de *ray marching* ou de *slicing* « classiques » à base de texture 3D, notre méthode devient rapidement pertinente lorsque la taille des données augmente et dépasse la capacité de stockage des GPU. Nous proposons également dans ce chapitre une discussion sur la notion de mésostructure pour un modèle volumique. Nous discutons en particulier de critères de décisions automatiques permettant la détéction des zones de détails pour un modèle volumique. Toutefois, cette méthode présente, comme le *splatting*, des performances moindres qu'une technique à base de textures 3D lorsque les données ne dépassent pas la capacité de stockage des GPU.
En résumé, cette méthode présente les avantages suivants :

- des performances accrues sur le *splatting*.
- un découplage « automatique ».
- une qualité visuelle importante.

Les limitations de cette méthodes peuvent se résumer ainsi :

- des performances moins efficaces que le *slicing* pour de « petits » modèles

Les points d'amélioration de cette méthode sont assez diverses. Par exemple, on peut considérer l'amélioration du schéma hiérarchique de simplification afin de mieux approcher les données. On peut également viser à obtenir une définition plus précise et moins empirique du critère de subdivision proposé, notamment en se basant sur le principe d'une décomposition des données en ondelettes.

9.1.3 Modélisation et rendu de mésostructures procédurales

Idée : Faciliter la création de mésostructures volumiques procédurales au travers d'une reformulation des hypertextures basée sur une fonction de transfert de forme permettant le contrôle intuitif de l'aspect visuel final.

Cette méthode, présentée au chapitre 7, propose l'ajout de détails visuels de mésostructures procédurales. Plus précisement, cette méthode se repose sur une reformulation du principe des hypertextures. Cette nouvelle approche permet principalement de faciliter le contrôle de l'utilisateur final sur le résultat visuel via l'utilisation d'une fonction de transfert de forme qui permet la spécification intuitive de manière graphique des détails de mésostructures. Au final, grâce à notre méthode de rendu hybride, nous proposons un outil interactif permettant l'exploration des possibilités d'une méthode procédurale d'ajout de détails volumiques. L'avantage principal de cette représentation consiste en son extrême compacité, sa continuité et son efficacité à représenter rapidement des phénomènes naturels présentant des mésostructures complexes. La souplesse de cette méthode permet également de réaliser facilement divers effets visuels au cours du temps, tel l'animation de la mésostructure, via une fonction de bruit dépendante du temps, ou encore l'effet de *morphing*, c'est-à-dire une transformation au cours du temps d'une mésostructure en une autre, via l'interpolation triviale entre deux fonctions de transfert.
Les avantages de notre méthode peuvent se résumer ainsi :

- un contrôle intuitif du résultat visuel,
- un retour visuel interactif,
- une animation du modèle possible.

Quelques limitations peuvent toutefois être citées :

- une difficulté d'un schéma adaptatif,
- un procédé manuel de création des fonctions de transfert de forme.

L'axe principal d'amélioration de cette représentation procédurale est la construction automatique de la fonction de transfert. En effet, il serait envisageable d'envisager, un peu à l'instar de la méthode proposée dans le chapitre 8, la synthèse des fonctions de transfert de forme en se basant sur l'analyse d'un exemple de mésostructure, issu d'une photo ou encore d'un modèle 3D très détaillé.

9.1.4 Génération de textures procédurales

Idée : Générer automatiquement à partir d'une texture d'exemple une description procédurale permettant la reconstruction au vol d'une texture « similaire » à la surface d'un modèle surfacique paramétré.

Finalement, nous avons présenté dans le chapitre 8, une méthode de génération procédurale de texture 2D en se basant sur l'analyse d'un exemple. Plus précisement, notre méthode propose de générer une description procédurale en vue de créer lors du rendu une apparence « similaire » à celle de l'image fournie en exemple. Cette méthode fonctionne par l'extraction des signaux aléatoires sous-jaçents ainsi que des distributions d'éléments visuels caractéristiques de la texture d'exemple et par la reproduction de ces signaux de manière fonctionnelle. La description générée est évalué pour chaque point de la surface lors de l'affichage.

Les avantages de notre description procédurale sont nombreux : les textures sont fonctionnellement définies dans un espace infini, ce qui permet de « texturer » à haute résolution de très grand modèles, le coût mémoire de cette représentation est quasi-nul, l'aspect visuel des textures peut être perturbé lors de l'affichage grâce aux nombreux paramètres disponibles.

Pour résumer, les avantages de notre méthode sont :

- une création automatique à partir d'une image d'exemple,
- une coût mémoire très faible,
- une grande résolution apparente,
- un contrôle de l'apaprence lors du rendu,
- une possibilité d'animation des textures.

Cet algorithme est toutefois sujet à quelques limitations :

- la limitation de la méthode à deux classes de textures seulement,
- l'impossibilité de reproduire tous les signaux aléatoires.

L'une des principales pistes d'amélioration de cette méthode consiste à accroître le champ d'application de cette méthode. Pour cela, il est nécessaire de lever la restriction imposée sur les classes de textures qu'il est possible de représenter. Pour cela, nous envisageons une technique reposant sur la déformation des motifs visuels en eux même, afin de faciliter la génération des textures quasi-régulières, procédurales, ou basées sur des motifs corrélés.

9.2 Perspectives

Nous concluons ce document par un ensemble des perspectives à plus long terme de nos travaux.

9.2.1 D'autres représentations de mésostructures

Une des pistes augmentant la portée de ce travail de thèse consiste naturellement à intégrer des techniques de rendu concernant différentes représentations de mésostructure, et plus particulièrement des représentations à base de primitives implicites. Des mésostructures définies implicitements présentent plusieurs avantages intéréssants :

- Une définition extrêmement compacte, permettant ainsi un stockage efficace sur le matériel graphique,
- Un calcul d'intersection par méthode de lancer de rayon efficace. Plusieurs travaux récents proposent l'implémentation matérielle du calcul de l'intersection d'un rayon de vue ave une primitive implicite, en se basant sur une méthode itérative [149] ou sur l'arithmétique affine des intervalles [**?**],
- Une précision bien supérieure aux définitions discrètes actuelles.

Le rendu d'une mésostructure bénéficiant de tels avantages semble prometeur et devrait permettre d'améliorer la qualité visuelle des images de synthèse. Toutefois, cette représentation est sujette à un inconvénient important : une difficulté de modélisation. En effet, le problème de la création de mésostructures arbitraires avec des primitives implicites reste difficile et limite encore l'utilisation d'une telle représentation.
Nous présentons sur la figure 9.1 quelques premiers résultats de nos travaux concernant le rendu de mésostructures définies implicitement. Ici, nous montrons deux exemples avec une mésostructure consistant en un tore illustrant la facilité ave laquelle il est possible d'ajuster les paramètres de répétition de la méthode ainsi que les performances (ici environ 30 fps) pour un grand nombre de primitives.

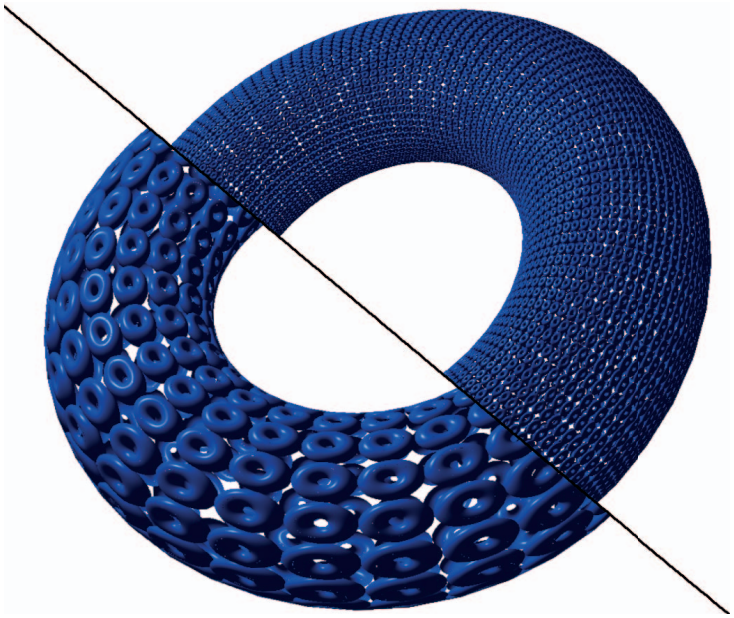


FIGURE 9.1 – Premiers travaux sur une mésostructure définie par un ensemble de primitives implicites ($30 fps$.

9.2.2 Génération procédurale de mésostructure complexe

Un des objectif idéal du domaine de l'information graphique est la visualisation d'environnements immenses visuellement riches à tous les niveaux de détails. La modélisation d'un tel environnement, outre le coût mémoire démesuré que représenterait une telle masse de données, demanderait à un créateur un temps démesuré. Un tel environnement bénéficierait d'une définition procédurale à tous niveaux, et particulièrement d'une génération procédurale basée sur un exemple et pourrait bénéficier d'une méthode de rendu hybride telle que celle présentée dans ces travaux.
Nous avons vu au travers de notre méthode de génération de textures semi-procédurales qu'il était possible d'appliquer des éléments visuels à la surface d'un modèle. Il est à notre sens possible de générer des informations plus complexes qu'une simple information de couleur, comme par exemple une carte de hauteur (en s'appuyant sur des techniques étudiant les représentations non-photoréalistes). Par exemple, ceci permettrait d'apposer de manière procédurale un élément « fenêtre » en relief à la surface d'un batîment de manière procédurale. Ces éléments de mésostructure peuvent également bénéficier, lorsque qu'ils sont proches de la caméra par exemple, de l'ajout de détails géométriques de mésostructure supplémentaire à plus petite échelle, matérialisant une sur-

face abîmée, des éraflures ou encore un phénomène d'érosion. Ces phénomènes peuvent eux aussi être générés de manière procédurale en se basant sur un ensemble d'exemples.
Par ce jeu de mésotructures à plusieurs niveaux, un ensemble de « briques » de base associé à un jeu de règles de distribution et de perturbation pourrait permettre la représentation réaliste et visuellement riche en détails à tous niveaux de visualisation d'un environnement immense avec un coût mémoire et un temps de modélisation extrêmement faible. Le développement rapide des performances des GPU et l'engouement actuel pour les méthodes basées images pourrait autoriser la visualisation d'un tel environnement en temps interactif.

9.2.3 Notion d'éclairage complexe

Un des points important pour le réalisme peu abordé dans cette thèse est le problème des interactions lumineuses complexes, comme l'ombrage et les inter-réflexions, qui sont aux final responsables de l'apparence de nos modèles. De la même manière que la visualisation de modèles complexes bénéficie d'un découpage en plusieurs niveaux structurels, les calculs d'illumination complexe pourrait également être accéléré par un tel découpage, comme le propose Sloan et al. [139]. Par exemple,le problème des inter-réflexions peut être traité plus facilement au niveau de la mésostructure, en négligeant les contributions de la macrostructure. De manière similaire, alors que les réflets spéculaires d'un modèle nécéssitent une grande précision, les interactions lumineuses diffuses peuvent être traitées à un niveau plus grossier. L'étude et le calcul des interactions lumineuses spécialement axées sur la mésostructure est également une piste important pour augmenter le réalisme des images de synthèse.

Bibliographie

[1] AL HAYTHAM, I. Kitab al-manazir (book of optics). iraq, 1021.

[2] ALEXA, M., BEHR, J., COHEN-OR, D., FLEISHMAN, S., LEVIN, D., AND SILVA, C. T. Point set surfaces. In *VIS 01: Proceedings of the conference on Visualization 01* (Washington, DC, USA, 2001), IEEE Computer Society, pp. 21–28.

[3] APPEL, A. Some techniques for shading machine renderings of solids. In *AFIPS 68 (Spring): Proceedings of the April 30–May 2, 1968, spring joint computer conference* (New York, NY, USA, 1968), ACM, pp. 37–45.

[4] ASHIKHMIN, M. Synthesizing natural textures. In *SI3D 01: Proceedings of the 2001 symposium on Interactive 3D graphics* (2001), ACM Press, pp. 217–226.

[5] ASHIKHMIN, M., AND SHIRLEY, P. An anisotropic phong brdf model. *J. Graph. Tools 5*, 2 (2000), 25–32.

[6] ASHIKHMIN, M., AND SHIRLEY, P. Steerable illumination textures. *ACM Trans. Graph. 21*, 1 (2002), 1–19.

[7] ASIRVATHAM, A., AND HOPPE, H. *GPU Gems 2 - Programming Techniques for High-Performance Graphics and General-Purpose Computation*. 2005, ch. Terrain rendering using GPU-based geometry clipmaps, pp. 27–45.

[8] A.V. OPPENHEIM, R.W. SCHAFER. *Digital signal processing*. Prentice-Hall, 1975.

[9] BABOUD, L., AND DÉCORET, X. Rendering geometry with relief textures. In *GI 06: Proceedings of Graphics Interface 2006* (Toronto, Ont., Canada, Canada, 2006), Canadian Information Processing Society, pp. 195–201.

[10] BADOUEL, D. An efficient ray-polygon intersection. 390–393.

[11] BAKER, B. S., GROSSE, E., AND RAFFERTY, C. S. Nonobtuse triangulation of polygons. *Discrete Comput. Geom. 3*, 2 (1988), 147–168.

[12] BAOQUAN LIU, GORDON J. CLAPWORTHY, F. D. Accelerating volume ray-casting using proxy spheres. In *EuroVis 2009* (2009).

[13] BAVOIL, L., CALLAHAN, S. P., LEFOHN, A., COMBA, J. A. L. D., AND SILVA, C. T. Multi-fragment effects on the gpu using the k-buffer. In *I3D 07: Proceedings of the 2007 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2007), ACM, pp. 97–104.

[14] BECHMANN, D., AND VERROUST-BLONDET, A. *Informatique Graphique, modélisation géométrique et animation.* Traité IC2. Hermès, march 2007, ch. Animation par déformations et métamorphoses, pp. 253–289.

[15] BLINN, J. Jim blinn's corner: Image compositing–theory. *IEEE Computer Graphics and Applications 14*, 5 (1994).

[16] BLINN, J. F. Simulation of wrinkled surfaces. *SIGGRAPH Comput. Graph. 12*, 3 (1978), 286–292.

[17] BLINN, J. F. Light reflection functions for simulation of clouds and dusty surfaces. *SIGGRAPH Comput. Graph. 16*, 3 (1982), 21–29.

[18] BLINN, J. F., AND NEWELL, M. E. Texture and reflection in computer generated images. *Commun. ACM 19*, 10 (1976), 542–547.

[19] BOTSCH M, SPERNAT M, K. L. Phong splatting. In *Proceedings of the symposium on point-based graphics* (2004), Eurographics Association, pp. 25–32.

[20] BOURQUE, E., AND DUDEK, G. Procedural texture matching and transformation. *Comput. Graph. Forum 23*, 3 (2004), 461–468.

[21] CABRAL, B., CAM, N., AND FORAN, J. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *VVS 94: Proceedings of the 1994 symposium on Volume visualization* (New York, NY, USA, 1994), ACM, pp. 91–98.

[22] CARR, N. A., HALL, J. D., AND HART, J. C. The ray engine. In *HWWS 02: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (Aire-la-Ville, Switzerland, Switzerland, 2002), Eurographics Association, pp. 37–46.

[23] CATMULL, E., AND CLARK, J. Recursively generated b-spline surfaces on arbitrary topological meshes. 183–188.

[24] CATMULL, E. E. *A subdivision algorithm for computer display of curved surfaces.* PhD thesis, 1974.

[25] CHEN, W., REN, L., ZWICKER, M., AND PFISTER, H. Hardware-accelerated adaptive ewa volume splatting. In *VIS 04: Proceedings of the conference on Visualization 04* (2004).

[26] CHEN, Y., TONG, X., WANG, J., LIN, S., GUO, B., AND SHUM, H.-Y. Shell texture functions. In *SIGGRAPH 04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 343–353.

[27] CIGNONI, P., MONTANI, C., SCOPIGNO, R., AND ROCCHINI, C. A general method for preserving attribute values on simplified meshes. In *VIS 98: Proceedings of the conference on Visualization 98* (Los Alamitos, CA, USA, 1998), IEEE Computer Society Press, pp. 59–66.

[28] CO, C. S., HAMANN, B., AND JOY, K. I. Iso-splatting: A point-based alternative to isosurface visualization. In *PG 03: Proceedings of the 11th Paci c Conference on Computer Graphics and Applications* (Washington, DC, USA, 2003), IEEE Computer Society, p. 325.

[29] COHEN, J., OLANO, M., AND MANOCHA, D. Appearance-preserving simplification. In *SIGGRAPH 98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), ACM, pp. 115–122.

[30] COOK, R. L. Shade trees. *SIGGRAPH Comput. Graph. 18*, 3 (1984), 223–231.

[31] COOK, R. L., CARPENTER, L., AND CATMULL, E. The reyes image rendering architecture. *SIGGRAPH Comput. Graph. 21*, 4 (1987), 95–102.

[32] COOK, R. L., AND TORRANCE, K. E. A reflectance model for computer graphics. *ACM Trans. Graph. 1*, 1 (1982), 7–24.

[33] CULLIP, T. J., AND NEUMANN, U. Accelerating volume reconstruction with 3d texture hardware. Tech. rep., Chapel Hill, NC, USA, 1994.

[34] DANA, K. J., VAN GINNEKEN, B., NAYAR, S. K., AND KOENDERINK, J. J. Reflectance and texture of real-world surfaces. *ACM Trans. Graph. 18*, 1 (1999), 1–34.

[35] DE TOLEDO, R., LEVY, B., AND PAUL, J.-C. Iterative methods for visualization of implicit surfaces on gpu. In *ISVC, International Symposium on Visual Computing* (Lake Tahoe, Nevada/California, November 2007), Lecture Notes in Computer Science, Springer, pp. 598–609.

[36] DECAUDIN, P., AND NEYRET, F. Volumetric billboards, 2009.

[37] DISCHLER, J., AND GHAZANFARPOUR, D. A geometrical based method for highly complex structured textures generation. *Computer Graphics forum 14*, 4 (1995), 203–215.

[38] DISCHLER, J., AND GHAZANFARPOUR, D. A procedural description of geometric textures by spectral and spatial analysis of profiles. *Computer Graphics forum (proc. of Eurographics 97) 16*, 3 (1997), 129–139.

[39] DISCHLER, J., AND GHAZANFARPOUR, D. A procedural description of geometric textures by spectral and spatial analysis of profiles. *Computer Graphics Forum 16*, 3 (1997), 129–139.

[40] DISCHLER, J., MARITAUD, K., LEVY, B., AND CHAZANFARPOUR, D. Texture particles. In *Eurographics 2002 - EG 2002, Saarbrücken, Germany* (Sep 2002).

[41] DISCHLER, J.-M. Efficiently rendering macro geometric surface structures with bi-directional texture functions. In *Eurographics Rendering Workshop 1998* (June 1998), pp. 169–180.

[42] DOO, D., AND SABIN, M. Behaviour of recursive division surfaces near extraordinary points. 177–181.

[43] DREBIN, R. A., CARPENTER, L., AND HANRAHAN, P. Volume rendering. In *SIGGRAPH 88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1988), ACM, pp. 65–74.

[44] EBERT, D., MUSGRAVE, K., PEACHEY, P., PERLIN, K., AND WORLEY, S. *Texturing and Modeling: A Procedural Approach*. 1998.

[45] EFROS, A. A., AND FREEMAN, W. T. Image quilting for texture synthesis and transfer. In *SIGGRAPH 01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM Press, pp. 341–346.

[46] ENGEL, K., KRAUS, M., AND ERTL, T. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *HWWS 01: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware* (2001).

[47] FABIO, P., AND M., O. M. *In ShaderX4: Advanced Rendering Techniques*. 2005, ch. Rendering surface details with relief mapping.

[48] FABIO, P., M., O. M., AND COMBA JO A. L. D. Real-time relief mapping on arbitrary polygonal surfaces. *ACM Trans. Graph. 24*, 3 (2005), 935–935.

[49] GAMITO, M., AND MADDOCK, S. Topological correction of hypertextured implicit surfaces for ray casting. *The Visual Computer 24* (2008), 397–409.

[50] GARDNER, G. Y. Simulation of natural scenes using textured quadric surfaces. *SIGGRAPH Comput. Graph. 18*, 3 (1984), 11–20.

[51] GARDNER, G. Y. Visual simulation of clouds. *SIGGRAPH Comput. Graph. 19*, 3 (1985), 297–304.

[52] GEOFF WYVILL, C. M., AND WYVILL, B. Data structure for soft objects. *The Visual Computer 2* (1986), 227–234.

[53] GERALD, C. F., AND WHEATLEY, P. O., Eds. *Applied numerical analysis*. Addison-Wesley Publishing Co, 1989.

[54] GHAZANFARPOUR, D., AND DISCHLER, J. Spectral analysis for automatic 3d texture generation. *Computers and Graphics 19*, 3 (1995), 413–422.

[55] GLASSNER, A. S., Ed. *An introduction to ray tracing*. Academic Press Ltd., London, UK, UK, 1989.

[56] GOODNIGHT, N., WOOLLEY, C., LEWIN, G., LUEBKE, D., AND HUMPHREYS, G. A multigrid solver for boundary value problems using programmable graphics hardware. In *HWWS 03: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 102–111.

[57] GROSSMAN, WILLIAM, GROSSMAN, J. P., AND DALLY, W. J. Point sample rendering. In *In Rendering Techniques Š98* (1998), Springer, pp. 181–192.

[58] GUSTAVSON, S. Simplex noise demystified. In *http://www.itn.liu.se/stegu/simplexnoise/* (March 2005).

[59] HANRAHAN, P. Ray tracing algebraic surfaces. In *SIGGRAPH 83: Proceedings of the 10th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1983), ACM, pp. 83–90.

[60] HEEGER, D. J., AND BERGEN, J. R. Pyramid-based texture analysis/synthesis. In *SIGGRAPH* (1995), pp. 229–238.

[61] HEIDRICH, W., DAUBERT, K., KAUTZ, J., AND SEIDEL, H.-P. Illuminating micro geometry based on precomputed visibility. In *SIGGRAPH 00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 455–464.

[62] JENSEN, H. W., MARSCHNER, S. R., LEVOY, M., AND HANRAHAN, P. A practical model for subsurface light transport. In *SIGGRAPH 01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 511–518.

[63] JULESZ, B. Textons, the elements of texture perception, and their interactions. *Nature 290*, 5802 (March 1981), 91–97.

[64] KAJIYA, J. T. The rendering equation. *SIGGRAPH Comput. Graph. 20*, 4 (1986), 143–150.

[65] KAJIYA, J. T., AND KAY, T. L. Rendering fur with three dimensional textures. In *SIGGRAPH 89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1989), ACM, pp. 271–280.

[66] KAJIYA, J. T., AND KAY, T. L. Rendering fur with three dimensional textures. *SIGGRAPH Comput. Graph. 23*, 3 (1989), 271–280.

[67] KAJIYA, J. T., AND VON HERZEN, B. P. Ray tracing volume densities. *SIGGRAPH Comput. Graph. 18*, 3 (1984), 165–174.

[68] KANEKO T., TAKAHEI T., I. M. K. N.-Y. Y. M. T. T. S. Detailed shape representation with parallax mapping. In *I3D 08: Proceedings of ICAT 2001* (2001), pp. 205–208.

[69] KAUTZ, J., SATTLER, M., SARLETTE, R., KLEIN, R., AND SEIDEL, H.-P. Decoupling brdfs from surface mesostructures. In *GI 04: Proceedings of Graphics Interface 2004* (School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004), Canadian Human-Computer Communications Society, pp. 177–182.

[70] KNOLL, A., HIJAZI, Y., WALD, I., HANSEN, C., AND HAGEN, H. Interactive ray tracing of arbitrary implicits with simd interval arithmetic. In *Proceedings of the 2nd IEEE/EG Symposium on Interactive Ray Tracing* (2007), pp. 11–18.

[71] KOPF, J., FU, C.-W., COHEN-OR, D., DEUSSEN, O., LISCHINSKI, D., AND WONG, T.-T. Solid texture synthesis from 2d exemplars. *ACM Trans. Graph. 26*, 3 (2007).

[72] KOUDELKA, M. L., AND MAGDA, S. Acquisition, compression, and synthesis of bidirectional texture functions. In *In Proc. 3rd Int. Workshop on Texture Analysis and Synthesis (Oct* (2003), pp. 59–64.

[73] KRAUS, M. Direct volume visualization of geometrically unpleasant meshes, 2003.

[74] KRAUS, M., STRENGERT, M., KLEIN, T., AND ERTL, T. Adaptive sampling in three dimensions for volume rendering on gpus. In *Visualization, 2007. APVIS 07. 2007 6th International Asia-Paci c Symposium on* (Feb. 2007), pp. 113–120.

[75] KRUEGER, W. The application of transport theory to visualization of 3d scalar data fields. In *VIS 90: Proceedings of the 1st conference on Visualization 90* (Los Alamitos, CA, USA, 1990), IEEE Computer Society Press, pp. 273–280.

[76] KRUGER, J., AND WESTERMANN, R. Acceleration techniques for gpu-based volume rendering. In *VIS 03: Proceedings of the 14th IEEE Visualization 2003 (VIS 03)* (Washington, DC, USA, 2003), IEEE Computer Society, p. 38.

[77] KRUGER, J., AND WESTERMANN, R. Acceleration techniques for gpu-based volume rendering. In *VIS 03: Proceedings of the 14th IEEE Visua-*

lization 2003 (VIS 03) (Washington, DC, USA, 2003), IEEE Computer Society, p. 38.

[78] KRÜGER, J., AND WESTERMANN, R. Linear algebra operators for gpu implementation of numerical algorithms. In *SIGGRAPH 03: ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), ACM, pp. 908–916.

[79] KRYACHKO, Y. *GPU Gems 2 - Programming Techniques for High-Performance Graphics and General-Purpose Computation.* 2005, ch. Using Vertex Texture Displacement for Realistic Water Rendering, pp. 283–294.

[80] KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics 22*, 3 (2003), 277–286.

[81] LAFORTUNE, E. P., AND WILLEMS, Y. D. Using the modified phong reflectance model for physically based rendering. Tech. rep., Department of Computing Science, K.U. Leuven, 1994.

[82] LAGAE, A., LEFEBVRE, S., DRETTAKIS, G., AND DUTRÉ, P. Procedural noise using sparse gabor convolution. *ACM Transactions on Graphics (SIGGRAPH Conference Proceedings) 28*, 3 (August 2009).

[83] LAMAR, E., HAMANN, B., AND JOY, K. I. Multiresolution techniques for interactive texture-based volume visualization. In *VIS 99: Proceedings of the conference on Visualization 99* (Los Alamitos, CA, USA, 1999), IEEE Computer Society Press, pp. 355–361.

[84] LAUR, D., AND HANRAHAN, P. Hierarchical splatting: a progressive refinement algorithm for volume rendering. In *SIGGRAPH 91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques* (1991).

[85] LEFEBVRE, S., AND HOPPE, H. Parallel controllable texture synthesis. *ACM Trans. Graph. 24*, 3 (2005), 777–786.

[86] LEFEBVRE, S., AND HOPPE, H. Appearance-space texture synthesis. In *SIGGRAPH 06: ACM SIGGRAPH 2006 Papers* (2006), pp. 541–548.

[87] LEFEBVRE, S., AND NEYRET, F. Pattern based procedural textures, 2003.

[88] LENSCH HENDRIK P. A., DAUBERT KATJA, S. H.-P. Interactive semi-transparent volumetric textures. In *Proceedings of Vision, Modeling, and Visualization VMV* (2002).

[89] LEVOY, M. Efficient ray tracing of volume data. *ACM Trans. Graph. 9*, 3 (1990), 245–261.

[90] LEVOY, M., AND WHITTED, T. The use of points as a display primitive. *Technical Report TR 85-022* (1985).

[91] LEWIS, J. Generalized stochastic subdivision. *ACM Trans. Graph. 6*, 3 (1987), 167–190.

[92] LEWIS, J. Algorithms for solid noise synthesis. In *Computer Graphics ACM Siggraph annual Conference Series* (1989), pp. 263–270.

[93] LIANG, L., LIU, C., XU, Y.-Q., GUO, B., AND SHUM, H.-Y. Real-time texture synthesis by patch-based sampling. *ACM Trans. Graph. 20*, 3 (2001), 127–150.

[94] LIU, Y., LIN, W.-C., AND HAYS, J. Near-regular texture analysis and manipulation. *ACM Trans. Graph. 23*, 3 (2004), 368–376.

[95] LIVNAT, Y., AND TRICOCHE, X. Interactive point-based isosurface extraction. In *VIS 04: Proceedings of the conference on Visualization 04* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 457–464.

[96] LIVNY, Y., BAUMAN, G., AND EL-SANA, J. Displacement patches for gpu-oriented view-dependent rendering. In *GRAPP* (2008), pp. 181–190.

[97] LIVNY, Y., KOGAN, Z., AND EL-SANA, J. Seamless patches for gpu-based terrain rendering. *Vis. Comput. 25*, 3 (2009), 197–208.

[98] LORENSEN, W. E., AND CLINE, H. E. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph. 21*, 4 (1987), 163–169.

[99] LÉVY, B. Constrained texture mapping. In *ACM SIGGRAPH conference proceedings* (Aug 2001), ACM, Addison Wesley.

[100] MA, K.-L., SCHUSSMAN, G., WILSON, B., KO, K., QIANG, J., AND RYNE, R. Advanced visualization technology for terascale particle accelerator simulations. In *Supercomputing 02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing* (2002), pp. 1–11.

[101] MALZBENDER, T., GELB, D., AND WOLTERS, H. Polynomial texture maps. In *SIGGRAPH 01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 519–528.

[102] MAMMEN, A. Transparency and antialiasing algorithms implemented with the virtual pixel maps technique. *IEEE Comput. Graph. Appl. 9*, 4 (1989), 43–55.

[103] MARSCHNER, S. R., AND LOBB, R. J. An evaluation of reconstruction filters for volume rendering. In *VIS 94: Proceedings of the conference on Visualization 94* (Los Alamitos, CA, USA, 1994), IEEE Computer Society Press, pp. 100–107.

[104] MAX, N. L. Horizon mapping: shadows for bump-mapped surfaces. *The Visual Computer 4*, 2 (July 1988), 109–117.

[105] MCGUIRE, M., AND MCGUIRE, M. Steep parallax mapping. *I3D 2005 Poster*.

[106] MEYER, A., AND NEYRET, F. Interactive volumetric textures, Jul 1998. ISBN.

[107] MILLER, C., AND JONES, M. Texturing and hypertexturing of volumetric objects. In *Volume Graphics* (2005), pp. 117–125.

[108] MÜLLER, G., MESETH, J., SATTLER, M., SARLETTE, R., AND KLEIN, R. Acquisition, synthesis and rendering of bidirectional texture functions. In *proceedings of Eurographics 2004, State of the Art Reports* (2004), pp. 69–94.

[109] MUELLER, K., SHAREEF, N., HUANG, J., AND CRAWFIS, R. High-quality splatting on rectilinear grids with efficient culling of occluded voxels. *IEEE Transactions on Visualization and Computer Graphics 5*, 2 (1999), 116–134.

[110] MUELLER, K., AND YAGEL, R. Fast perspective volume rendering with splatting by using a ray-driven approach. In *IEEE Visualization 96* (1996).

[111] MÜLLER, G., MESETH, J., AND KLEIN, R. Compression and real-time rendering of measured btfs using local pca.

[112] MÜLLER, P., ZENG, G., WONKA, P., AND GOOL, L. V. Image-based procedural modeling of facades. *ACM Trans. Graph. 26*, 3 (2007), 85.

[113] NEOPHYTOU, N., AND MUELLER, K. Post-convolved splatting. In *Joint Eurographics - IEEE TCVG Symposium on Visualization 2003* (2003).

[114] NEOPHYTOU, N., MUELLER, K., MCDONNELL, K. T., HONG, W., GUAN, X., QIN, H., AND KAUFMAN, A. Gpu-accelerated volume splatting with elliptical rbfs. In *IEEE TCVG Symposium on Visualization 2006*.

[115] NICODEMUS, F. E., RICHMOND, J. C., HSIA, J. J., GINSBERG, I. W., AND LIMPERIS, T. Geometrical considerations and nomenclature for reflectance. 94–145.

[116] OLIVEIRA, M. M., BISHOP, G., AND MCALLISTER, D. Relief texture mapping. In *SIGGRAPH 00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 359–368.

[117] OLIVEIRA M., P. F. An efficient representation for surface details. In *Tech. rep.,2005, UFRGS Technical Report RP-351*.

[118] PEACHEY, D. R. Solid texturing of complex surfaces. In *SIGGRAPH 85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1985), ACM, pp. 279–286.

[119] PENG, J., KRISTJANSSON, D., AND ZORIN, D. Interactive modeling of topologically complex geometric detail. In *SIGGRAPH 04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 635–643.

[120] PERLIN, K. An image synthesizer. In *Computer Graphics ACM Siggraph annual Conference Series* (1985), pp. 287–296.

[121] PERLIN, K. Noise hardware. In *Real-Time Shading SIGGRAPH Course Notes* ((Ed.), 2001), Olano M.

[122] PERLIN, K. Improving noise. *ACM Trans. Graph. 21*, 3 (2002), 681–682.

[123] PERLIN, K., AND HOFFERT, E. Hypertextures. In *Computer Graphics ACM Siggraph annual Conference Series* (1989), vol. 23, pp. 253–262.

[124] PFISTER, H., ZWICKER, M., VAN BAAR, J., AND GROSS, M. Surfels: surface elements as rendering primitives. In *SIGGRAPH 00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 335–342.

[125] PHONG, B. T. Illumination for computer generated pictures. *Commun. ACM 18*, 6 (1975), 311–317.

[126] POLICARPO, F., AND OLIVEIRA, M. M. Relief mapping of non-height-field surface details. In *I3D 06: Proceedings of the 2006 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2006), ACM, pp. 55–62.

[127] PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. Lapped textures. In *SIGGRAPH 00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 465–470.

[128] PURCELL, T. J., BUCK, I., MARK, W. R., AND HANRAHAN, P. Ray tracing on programmable graphics hardware. In *SIGGRAPH 02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM, pp. 703–712.

[129] RALF KAEHLER, T. A., AND HEGE, H.-C. Simultaneous gpu-assisted raycasting of unstructured point sets and volumetric grid data. In *Eurographics / IEEE VGTC Workshop on Volume Graphics 2007 (2007)* (2007).

[130] REEVES, W. T. Particle systems—a technique for modeling a class of fuzzy objects. *ACM Trans. Graph. 2*, 2 (1983), 91–108.

[131] REN, L., PFISTER, H., AND ZWICKER, M. Object space ewa surface splatting: A hardware accelerated approach to high quality point rendering, 2002.

[132] RESHETOV, A., SOUPIKOV, A., AND HURLEY, J. Multi-level ray tracing algorithm. In *SIGGRAPH 05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM, pp. 1176–1185.

[133] ROETTGER, S., GUTHE, S., WEISKOPF, D., ERTL, T., AND STRASSER, W. Smart hardware-accelerated volume rendering. In *Proc. of VisSym 03* (2003).

[134] ROETTGER, S., KRAUS, M., AND ERTL, T. Hardware-accelerated volume and isosurface rendering based on cell-projection. In *VISUALIZATION 00: Proceedings of the 11th IEEE Visualization 2000 Conference (VIS 2000)* (Washington, DC, USA, 2000), IEEE Computer Society.

[135] ROGER, D., ASSARSSON, U., AND HOLZSCHUCH, N. Whitted ray-tracing for dynamic scenes using a ray-space hierarchy on the gpu, jun 2007.

[136] RUSINKIEWICZ, S., AND LEVOY, M. Qsplat: a multiresolution point rendering system for large meshes. In *SIGGRAPH 00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 343–352.

[137] SABELLA, P. A rendering algorithm for visualizing 3d scalar fields. In *SIGGRAPH 88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1988), ACM, pp. 51–58.

[138] SATHERLEY, R., AND JONES, M. Hypertexturing complex volume objects. *The Visual Computer 18* (2002), 226–235.

[139] SLOAN, P.-P., LIU, X., SHUM, H.-Y., AND SNYDER, J. Bi-scale radiance transfer. *ACM Trans. Graph. 22*, 3 (2003), 370–375.

[140] SLOAN, P.-P. J., AND COHEN, M. F. Interactive horizon mapping. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000* (London, UK, 2000), Springer-Verlag, pp. 281–286.

[141] SONG, Y., CHEN, Y., TONG, X., LIN, S., SHI, J., GUO, B., AND SHUM, H.-Y. Shell radiance texture functions. *The Visual Computer 21*, 8.

[142] STEGMAIER, S., STRENGERT, M., KLEIN, T., AND ERTL, T. A simple and flexible volume rendering framework for graphics-hardware-based raycasting. In *Volume Graphics, 2005. Fourth International Workshop on* (June 2005), pp. 187–241.

[143] STEWART, A. J. Vicinity shading for enhanced perception of volumetric data. In *VIS 03: Proceedings of the 14th IEEE Visualization 2003 (VIS 03)* (Washington, DC, USA, 2003), IEEE Computer Society, p. 47.

[144] STRNAD, D., AND GUID, N. Modeling trees with hypertextures. *Computer Graphics Forum 23*, 2 (2004), 173–187.

[145] SUN, M. D. Data complexity for virtual reality: Where do all the triangles go? In *In IEEE Virtual Reality Annual International Symposium (VRAIS* (1994), pp. 357–363.

[146] SZIRMAY-KALOS, L., AND UMENHOFFER, T. Displacement mapping on the GPU - State of the Art. *Computer Graphics Forum 27*, 1 (2008).

[147] T., W. Parallax mapping with offset limiting:a perpixel approximation of uneven surfaces. *Tech. rep.,In scape Corporation* (2004).

[148] TOLEDO, R. *Visualisation interactive de modeles complexes en utilisant les cartes graphiques*. PhD thesis, INPL, France, october 2007.

[149] TOLEDO, R. D., WANG, B., AND LEVY, B. Geometry textures. In *SIBGRAPI 07: Proceedings of the XX Brazilian Symposium on Computer Graphics and Image Processing* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 79–86.

[150] TOTH, D. L. On ray tracing parametric surfaces. In *SIGGRAPH 85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1985), ACM, pp. 171–179.

[151] TURK, G. Texture synthesis on surfaces. In *SIGGRAPH 01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM Press, pp. 347–354.

[152] TUY, H., AND TUY, L. Direct 2d display of 3d objects. *IEEE Computer Graphics and Applications 4*, 10 (1984), 2933.

[153] VAN WIJK, J. J. Ray tracing objects defined by sweeping a sphere. In *Eurographics 84* (1984), pp. 73–82.

[154] VASILESCU, M. A. O., AND TERZOPOULOS, D. Tensortextures: multilinear image-based rendering. *ACM Trans. Graph. 23*, 3 (2004), 336–342.

[155] WALD, I., SLUSALLEK, P., BENTHIN, C., AND WAGNER, M. Interactive rendering with coherent ray tracing. In *Computer Graphics Forum* (2001), pp. 153–164.

[156] WANG, L., WANG, X., TONG, X., LIN, S., HU, S., GUO, B., AND SHUM, H.-Y. View-dependent displacement mapping. *ACM Trans. Graph. 22*, 3 (2003), 334–339.

[157] WANG X., TONG X., L. S. H. S.-G. B., AND H.-Y., S. Generalized displacement maps. In *Eurographics Symposium on Rendering* (2003), pp. 228–233.

[158] WARD, G. J. Measuring and modeling anisotropic reflection. *SIGGRAPH Comput. Graph. 26*, 2 (1992), 265–272.

[159] WEI, L.-Y., LEFEBVRE, S., KWATRA, V., AND TURK, G. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR* (2009), Eurographics Association.

[160] WEI, L.-Y., AND LEVOY, M. Fast texture synthesis using tree-structured vector quantization. In *SIGGRAPH 00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 479–488.

[161] WEI, L.-Y., AND LEVOY, M. Texture synthesis over arbitrary manifold surfaces. In *SIGGRAPH 2001, Computer Graphics Proceedings* (2001), E. Fiume, Ed., ACM Press / ACM SIGGRAPH, pp. 355–360.

[162] WESTIN, S. H., ARVO, J. R., AND TORRANCE, K. E. Predicting reflectance functions from complex surfaces. *SIGGRAPH Comput. Graph. 26*, 2 (1992), 255–264.

[163] WESTOVER, L. Interactive volume rendering. In *VVS 89: Proceedings of the 1989 Chapel Hill workshop on Volume visualization* (1989), pp. 9–16.

[164] WESTOVER, L. Footprint evaluation for volume rendering. In *SIGGRAPH 90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1990), ACM, pp. 367–376.

[165] WHITTED, T. An improved illumination model for shaded display. *Commun. ACM 23*, 6 (1980), 343–349.

[166] WILLIAMS, L. Casting curved shadows on curved surfaces. In *SIGGRAPH 78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1978), ACM, pp. 270–274.

[167] WILLIAMS, P. L., MAX, N. L., AND STEIN, C. M. A high accuracy volume renderer for unstructured data. *IEEE Transactions on Visualization and Computer Graphics 4* (1998), 37–54.

[168] WILSON, B., MA, K.-L., AND MCCORMICK, P. S. A hardware-assisted hybrid rendering technique for interactive volume visualization. In *VVS 02: Proceedings of the 2002 IEEE symposium on Volume visualization and graphics* (2002).

[169] WORLEY, S. A cellular texturing basis function. In *Computer Graphics ACM Siggraph annual Conference Series* (1996), pp. 291–294.

[170] WORLEY, S., AND HART, J. Hyper-rendering of hyper-textured surfaces. In *Implicit Surfaces 96 Eindhoven (The Netherlands)* (1996).

[171] XUE, D., AND CRAWFIS, R. Efficient splatting using modern graphics hardware. In *Journal of Graphics Tools* (2003), vol. 8, pp. 1–21.

[172] YING, L., HERTZMANN, A., BIERMANN, H., AND ZORIN, D. Texture and shape synthesis on surfaces. In *Proceedings of the 12th Eurographics*

Workshop on Rendering Techniques (2001), S. J. Gortler and K. Myszkowski, Eds., Springer, pp. 301–312.

[173] ZELINKA, S., AND GARLAND, M. Jump map-based interactive texture synthesis. *ACM Trans. on Graph. 23*, 4 (2004), 930–962.

[174] ZHANG, J., ZHOU, K., VELHO, L., GUO, B., AND SHUM, H.-Y. Synthesis of progressively-variant textures on arbitrary surfaces. *ACM Trans. Graph. 22*, 3 (2003), 295–302.

[175] ZWICKER, M., PFISTER, H., VAN BAAR, J., AND GROSS, M. Ewa volume splatting. In *VIS 01: Proceedings of the conference on Visualization 01* (2001), pp. 29–36.

[176] ZWICKER, M., PFISTER, H., VAN BAAR, J., AND GROSS, M. Surface splatting. In *SIGGRAPH 01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM, pp. 371–378.

[177] ZWICKER, M., PFISTER, H., VAN BAAR, J., AND GROSS, M. Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics 8*, 3 (2002), 223–238.