Université de Limoges

ÉCOLE DOCTORALE « Sciences et Ingénierie pour l'Information, Mathématiques »
FACULTÉ DES SCIENCES ET TECHNIQUES
Laboratoire XLIM - DMI

THÈSE

pour obtenir le grade de
DOCTEUR DE L'UNIVERSITÉ DE LIMOGES
Discipline / Spécialité : Informatique Graphique
présentée et soutenue par
Nicolas Pavie
le 3 novembre 2016

Modélisation par bruit procédural et rendu de détails volumiques de surfaces dans les scènes virtuelles

Thèse dirigée par Djamchid Ghazanfarpour,
Co-encadrée par Guillaume Gilet

JURY

Rapporteurs :

M. Kadi BOUATOUCH	Professeur de l'Université de Rennes 1
M. Yves DUTHEN	Professeur de l'Université de Toulouse 1 Capitole

Examinateurs :

M. Jean-Michel DISCHLER	Professeur de l'Université de Strasbourg
M. Djamchid GHAZANFARPOUR	Professeur de l'Université de Limoges
M. Guillaume GILET	Maître de Conférences de l'Université de Limoges

In creating, the only hard thing's to begin :
a grass blade's no easier to make than an oak.
James Russel Lowel, A Fable for Critics.

À Cécile.

Résumé

L'augmentation de la puissance graphique des ordinateurs grands publics entraîne avec elle une demande croissante de qualité et de complexité des scènes virtuelles. La gestion de cette complexité est particulièrement difficile pour les objets naturels tels les arbres et les champs d'herbe ou encore pour les animaux, pour lesquels de très nombreux petits objets très similaires viennent décorer les surfaces. La diversité de ces détails de surfaces, nécessaire à un rendu réaliste dans le cas des objets naturels, se traduit par une augmentation du temps de modélisation, du coût en stockage et de la complexité d'évaluation.

Nous nous sommes intéressés aux différentes représentations et méthodes de génération à la volée pouvant être utilisées pour la création et le rendu temps réel de ces détails sur de vastes surfaces. Nous avons concentré notre étude sur le cas particulier des champs d'herbe et des fourrures : De nombreux brins quasi-similaires, distribués aléatoirement sur la surface, forment une apparence visuelle très proche d'un motif de bruit incluant des éléments de structure.

Nous présentons dans un premier temps un bruit procédural axé sur la modélisation spatiale interactive d'éléments quasi-similaires et de leur distribution. L'utilisation de fonctions gaussiennes elliptiques comme primitive de modélisation, et la distribution non-uniforme contrôlée des éléments créés, permet de produire des motifs aléatoires ou quasi-réguliers incluant des caractéristiques structurelles. Une méthode d'analyse par décomposition en ellipses permet de préconfigurer ce bruit pour une reproduction rapide d'un motif donné.

Nous présentons ensuite une extension de ce bruit pour la modélisation procédurale d'une surcouche volumique composée de détails de surfaces tels que des brins ou des objets volumiques plus complexes. Pour conserver une modélisation interactive du motif, une première méthode de rendu d'ordre image et une seconde méthode d'ordre objet sont proposées pour une évaluation optimisée du bruit par une carte graphique. Ces deux méthodes permettent une visualisation interactive et visuellement convaincante du résultat.

Mots clefs : Informatique Graphique, textures procédurales, bruits procéduraux, rendu volumétrique, décoration de surfaces.

Abstract

The growing power of graphics processing units (GPU) in mainstream computers creates a need for a higher quality and complexity of virtual scenes. Managing this complexity for natural objects such as trees or grass fields or even animals is painstaking, due to the large amount of small objects decorating their surface. The diversity of such details, mandatory for realistic rendering of natural objects, translates in a longer authoring time, a higher memory requirement and a more complex evaluation.

We review in this thesis the related works on data representations and on-the-fly generation methods used for the creation and real-time rendering of details over large surfaces. We focus our study on the particular case of grass fields and fur : the fuzzy visual appearance of those surfaces is obtained by the distribution of many self-similar blades or strands, creating a pattern closely related to a noise with structural features.

We first present a procedural noise that aims at spatial modeling of self-similar elements and their distribution. The elliptical Gaussian function used as a modeling primitive and the controlled non-uniform distribution of elements allows for various type of patterns to be modeled, from stochastic to near-regular one, while including structural features. The by-example analysis process based on an ellipse fitting method allows a fast configuration of the noise for patterns reproduction.

We further introduce an extension of this noise model for the authoring of procedural shell textures of strand-based or more complex volumetric details. For interactive authoring of such volumetric pattern, an image-order and an object-order rendering methods are proposed, both methods being optimized for an implementation on the GPU. Our rendering methods allow for interactive visualization of a visually-convincing result.

Keywords: Computer Graphics, procedural texturing, procedural noise, volumetric rendering, surface decoration.

Table des matières

Table des figures

Introduction

L'informatique graphique connaît un intérêt croissant pour ses applications possibles dans de nombreux domaines. Les industries du divertissement, telles que le cinéma ou le jeu vidéo, voient dans les avancées du domaine de nouvelles possibilités d'immersion du spectateur dans des mondes virtuels. De nombreux studios modélisent dans ce sens des scènes virtuelles de plus en plus complexes, tentant de générer par ordinateurs des environnements se rapprochant le plus possible d'un visuel réaliste pour favoriser cette immersion. Avec l'évolution du matériel graphique grand public, nombre d'utilisateurs souhaitent pouvoir explorer ces scènes complexes librement et interactivement. Ces évolutions matérielles permettent la visualisation en temps réel d'univers toujours plus grands et composés d'objets toujours plus détaillés.

La création de grandes scènes virtuelles telles que les paysages est généralement découpée en différentes étapes, correspondant aux niveaux d'échelle de visualisation planifiés. La scène est d'abord constituée d'éléments macroscopiques tels que le terrain et les bâtiments, sur lesquels sont ensuite placés des éléments de plus en plus petits : les arbres, les fleurs, les feuilles etc. Les détails infiniment petits dans l'image sont finalement conçus comme des matériaux de surface ou des textures, décrivant le comportement de la lumière réfléchie par ces objets. Les petits objets ou détails de surface peuvent rendre la modélisation et la visualisation multi-échelles de la scène particulièrement complexe. Prenant comme exemple les champs d'herbe, un observateur perçoit très distinctement les différents brins proches de lui, mais ces éléments tendent à former une texture de surface au fur et à mesure que son regard se lève vers l'horizon, et cela sans discontinuités perceptibles. Une autre difficulté, posée plus généralement par la décoration d'objets naturels et organiques, est la présence nécessaire au réalisme de nombreuses variations ou déformations locales. La surface des objets naturels réel est rarement lisse et sans défaut, et ceux pour de nombreux types de surface (figure I) : Ces objets sont constamment modifiés par des interactions avec leur environnement sur différentes échelles de temps. De nombreux phénomènes physiques, chimiques et/ou biologique tels que le vent, l'humidité, ou encore l'illumination prolongée de certaines partie de la surface peuvent ainsi sensiblement modifier l'apparence finale de deux objets initialement identiques.

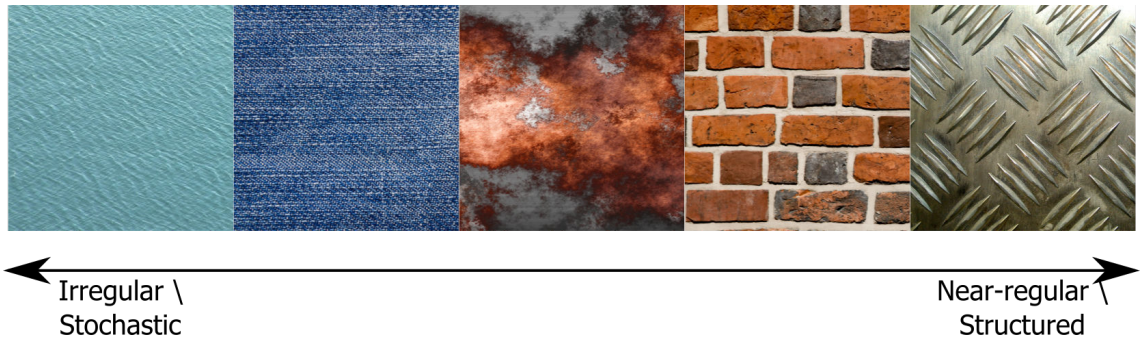


FIGURE I – Différents types de texture d'objets réels.

S'il est possible pour un artiste de créer manuellement ces détails variés, le temps de modélisation devient rapidement prohibitif avec l'augmentation de la taille de la surface et, par extension, l'augmentation de la quantité de différents détails à modéliser. Une autre approche possible, et utilisée dans de nombreux logiciels de modélisation scientifiques et industriels, repose sur l'utilisation de méthodes procédurales reproduisant les phénomènes cités précédemment pour générer un résultat réaliste. Ces phénomènes sont exprimés sous la forme de modèles mathématiques ou informatiques pouvant être évalués par différentes méthodes de simulation. Ces dernières peuvent s'inspirer de la physique (tels que les systèmes de particules), de l'intelligence artificielle (tels que les systèmes multi-agents), ou reposer sur des grammaires génératives (tels que les L-systèmes). Le résultat produit dépend alors des paramètres donnés en entrée de la méthode utilisée. Les logiciels de modélisation privilégient cependant la qualité de production plutôt que la vitesse de génération : la production d'un résultat par ces méthodes peut nécessiter un temps d'évaluation conséquent selon la quantité de détails ciblée.

Le résultat obtenu est souvent exprimé en une représentation dépendante du type de détails modélisé : les détails de type « textures » tels que ceux présentés dans la figure I sont souvent exprimés sous la forme d'images plaquées sur des surfaces, tandis que les petits objets tels que les brins d'herbes ou de fourrure (figure II) sont plus couramment modélisés sous une forme géométrique. Cette géométrie peut devenir un facteur de limitation des performances pour le rendu temps réel de la scène finale. Ces très nombreux éléments géométriques (pouvant dépasser la centaine de millions d'objets dans une image) peuvent représenter une quantité de mémoire surpassant celle d'une carte graphique et surpassant ses capacités de traitements simultanés de géométries ou de pixels. De nombreux artefacts de transparence et des escaliers de pixels peuvent également apparaître sans l'utilisation d'un ordonnancement des primitives rendues et un schéma complexe de filtrage de la géométrie. Pour palier ces problématiques, ces détails peuvent être transformés en d'autres représentations discrètes selon leurs différentes échelles de visualisation possibles (figure III). Ils peuvent par exemple être transformés en images ou en volumes distribués sur la surface de l'objet pour approcher l'apparence visuelle attendue à longue ou moyenne distance de la caméra. Ces différentes représentations permettent de réduire la complexité de rendu par un compromis sur la qualité visuelle des détails. Les représentations discrètes restent cependant limitées à une résolution visuelle maximale fixée, et la quantité maximale de détails visuels stockés dépend de la mémoire disponible.

(a) Une fourrure vue de près

(b) Un champ d'herbes vu de près

FIGURE II – Des motifs naturels composés de nombreux éléments quasi-similaires.

Une troisième approche est possible pour la génération des détails évoqués jusqu'ici : Bien que résultant de phénomènes naturels très variés, chacun de ces détails produit finalement un motif dans lequel ces variations peuvent sembler visuellement aléatoires. Certaines de ces variations peuvent être efficacement approchées par une modélisation statistique selon l'échelle visuelle ciblée. L'utilisation d'une telle modélisation permet une génération beaucoup plus rapide, possiblement réalisable à la volée (c'est-à-dire lors du rendu final), de détails très similaires aux méthodes précédentes. Pour réaliser cette approximation, les méthodes de générations à la volée peuvent se baser sur une distribution procédurale aléatoire de modèles utilisant les représentations discrètes précédentes, ou bien directement générer des éléments lors du rendu, en intégrant des variations aléatoires par élément. Les évolutions des cartes graphiques, telles que l'apparition des unités de tesselations ou la puissance grandissante des unités de traitement par pixel, permettent d'implanter des méthodes de génération procédurale de géométrie ou des textures procédurales évaluables en temps réel. Ces dernières permettent de remplacer des images ou des volumes discrets par un programme évaluable en tout point quelconque d'un espace de manière continue et indépendamment d'une quelconque précédente évaluation précédente. Certaines de ces textures procédurales se basent sur des motifs aléatoires générés par des bruits procéduraux, et se spécialisent dans la production de de textures non-structurées ou faiblement structurées particulières.

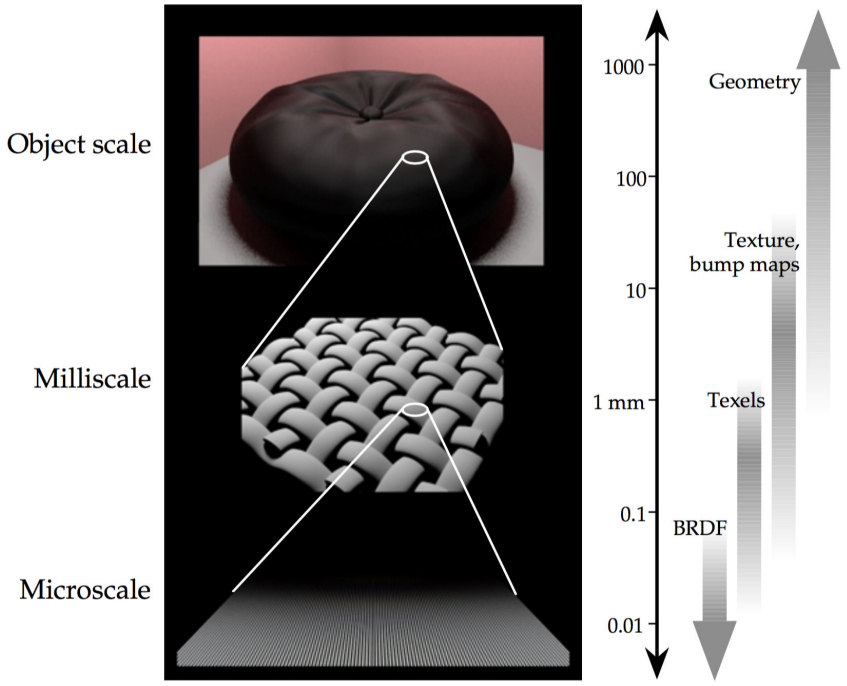


FIGURE III – Echelle d'application des représentations par Westin [WAT92].

Bien qu'offrant de nombreux avantages pour la création de motifs aléatoires, des problèmes persistent quant à l'utilisation de ces bruits pour la modélisation interactive de motifs stochastiques comprenant des éléments de structure. Les paramètres de ces fonctions n'interviennent pas directement dans le visuel du motif produit, mais dans la forme du spectre de puissance sous-jacent. L'édition d'un motif à travers ce spectre est particulièrement complexe, la corrélation entre espace fréquentiel et spatial (c'est-à-dire l'apparence visuelle) n'étant pas intuitive. L'édition du seul spectre de puissance limite également les possibilités de rajout d'éléments structurels complexes dans le motif. De nombreuses fonctions de bruit proposent dans ce sens des processus d'analyse et d'extraction automatique des paramètres contrôlant ce spectre à partir d'un exemple, mais le motif produit reste uniquement éditable interactivement par paramétrage du spectre de puissance.

L'utilisation d'une texture procédurale volumique pose également un problème de précision et de rapidité d'évaluation pour le rendu temps réel de détails de types « brins » tels que ceux de la figure II. L'évaluation d'une texture procédurale volumique est généralement réalisée par un échantillonnage pas à pas du transport de la lumière dans ce volume. Pour un rendu réaliste de brins volumiques, un pas d'échantillonnage extrêmement petit est nécessaire, réduisant d'autant les performances de rendu.

Les travaux présentés dans cette thèse associent un bruit procédural à différentes méthodes de rendu volumétriques optimisées pour la génération rapide de détails de surface complexes. Nous proposons dans un premier temps un modèle de bruit générique permettant l'édition spatiale d'éléments structurels dans des motifs stochastiques. Ce modèle facilite la modélisation interactive de motifs aléatoires semi-structurées, mais peut également créer des motifs quasi-structurées grâce à la définition d'une distribution aléatoire non-uniforme des éléments. Cette dernière permet un contrôle interactif de la répartition périodique ou apériodique des éléments structurels dans le résultat. Une méthode d'extraction automatique des paramètres est également proposée pour simplifier la configuration (ou pré-configuration) du bruit procédural. Nous proposons dans un second temps une version spécifique de ce modèle de bruit pour la modélisation de détails de surface dans une surcouche volumique. Une évaluation optimisée des éléments composant ce bruit est proposée puis intégrée dans des méthodes de rendu volumétrique d'ordre image et d'ordre objet. Ces différentes méthodes nous permettent une visualisation interactive de motifs volumiques complexes, nous rapprochant de notre objectif final : la génération et le rendu en temps réel de très nombreux détails volumiques de surfaces.

Organisation du document

Le premier chapitre de ce mémoire présente les travaux en relation avec nos recherches. Nous consacrons une première partie du chapitre au rappel des différentes représentations possibles pour le stockage et le rendu de détails de surface similaires à ceux ciblés par nos travaux. Une deuxième partie de ce chapitre récapitule les différentes méthodes existantes pour la génération procédurale de ce type de détails à la volée. À partir de ces différentes recherches, Nous avons choisi d'utiliser une texture procédurale générée à l'aide d'un bruit procédural dans une surcouche volumique. Le troisième partie de ce chapitre fournit un état de l'art des différents bruits procéduraux existants, prérequis nécessaire pour une meilleure compréhension de notre modèle de bruit. La dernière partie de ce chapitre rappelle la théorie fondamentale du transport de la lumière dans un volume et les méthodes de rendu volumétrique existantes, sur lesquelles se basent nos travaux sur la visualisation de ce bruit évalué dans une surcouche volumique.

Le second chapitre est dédié à notre première contribution, à savoir notre nouveau modèle de bruit procédural. Nous présentons dans ce chapitre une méthode de construction d'un bruit procédural à l'aide de fonctions gaussiennes elliptiques, ces dernières servant de primitives de modélisation et pouvant créer des éléments structurels dans un motif stochastique. Ce bruit présente également la particularité d'utiliser une distribution non-uniforme, permettant de créer des motifs allant de complètement stochastiques à semi-réguliers.

Le troisième chapitre présente une extension de ce modèle de bruit pour modéliser et visualiser des détails de surface 3D dans une surcouche volumique. Nous présentons une formulation utilisant un noyau de convolution basé sur une somme de gaussiennes ellipsoïdales filtrées permettant d'obtenir un rendu anti-aliasée de ces détails. Les méthodes de rendu volumétrique temps réel d'ordre objet ou d'ordre image standards pouvant créer de nombreuses redondances d'évaluation ou des artefacts visuels, nous introduisons une évaluation optimisée des noyaux par un échantillonnage pas à pas limité du rayon. Nous utilisons cette évaluation dans deux méthodes de rendu volumétrique optimisées pour la visualisation de notre bruit et implantées sur carte graphique : Une méthode d'ordre image basée sur un échantillonnage optimisé de l'espace d'évaluation, et une méthode d'ordre objet basée sur la projection et la rastérisation des noyaux.

Nous concluons enfin sur l'analyse de l'apport de ces contributions, en rappelant leurs limites et les possibles améliorations pouvant être apportées à notre modèle de bruit et à nos méthodes de rendu.

Chapitre 1

État de l'Art

Sommaire

Chapitre 1

État de l'Art

Une approche privilégiée pour la conception d'une scène virtuelle consiste à créer les éléments de la scène des plus grands éléments (terrain, bâtiments, arbres) aux plus petits (feuilles, fleurs, herbes). Dans les scènes de végétation, les petits éléments de surface tels que les brins ou les feuilles ont un impact majeur sur le réalisme des objets sur lesquels ils sont distribués, et par extension sur le réalisme de la scène. Mais ces petits éléments de surface sont particulièrement difficiles à concevoir manuellement pour un artiste et à visualiser en temps réel : les brins d'herbe, les fleurs ou encore les feuilles peuvent être présents en très grand nombre sur le terrain ou les arbres, et peuvent être de formes et d'orientations très variées. Dans ce chapitre, nous nous intéressons aux travaux existants sur les représentations pour le rendu temps réel et les méthodes de génération à la volée de ces détails de surface 3D complexes.

Pour optimiser la complexité ou la qualité du résultat affiché, les artistes et concepteurs ont la possibilité d'utiliser différentes représentations de ces détails. Ils peuvent les modéliser sous la forme de surfaces, sous la forme d'une image représentant l'apparence attendue à l'écran, ou sous la forme d'un volume. Prenant comme cas d'étude les détails de surface de type brins d'herbe et fourrures, nous rappelons ces représentations dans la section 1.1. En plus de ces représentations, les artistes et les concepteurs peuvent utiliser des méthodes procédurales de génération à la volée, pour faciliter la modélisation et réduire la quantité de mémoire nécessaire au stockage de ces détails. Nous rappelons ces différentes méthodes dans la section 1.2.

La solution envisagée à travers cette thèse est une représentation de surcouche volumique dans laquelle est évalué un bruit procédural générant ces détails. Pour introduire les prérequis nécessaire à une meilleur compréhension de nos contributions, nous rappelons d'abord dans la section 1.3 les différents bruits procéduraux existants, ainsi que leurs propriétés spatiales et fréquentielles. Nous rappelons finalement dans la section 1.4 les notions théoriques liées au rendu volumétrique, et les méthodes de rendu temps réels existantes pour les représentations volumiques.

1.1 Les représentations pour le rendu temps réel de brins

La qualité d'une scène de végétation est très fortement liée au niveau de finition apporté aux objets naturels, et en particulier aux décorations et aux altérations subies par les surfaces en différents endroits. Ces finitions peuvent être des déformations de la surface d'un objet, ou des petits objets distribués sur cette surface. Les petits objets tels que les brins d'herbe, présents en très grande quantité par zone de surface, forment une surface à part entière vus de loin. La véritable surface de l'objet sur laquelle sont situés ces éléments ne se révèle que lorsque l'observateur se rapproche. Le réalisme visuel de nombreux objets est lié en particulier à la présence d'un très grand nombre de petits objets similaires mais ayant subi des variations liées à des phénomènes naturels au cours du temps. Pour les objets tels que les poils d'une fourrure, les brins d'un champ d'herbe ou encore les feuilles sur les branches des arbres, certains de ces phénomènes peuvent être approximés statistiquement en introduisant différentes variantes d'un même modèle ayant subi des variations aléatoires.

Ces petits objets peuvent être directement intégrés géométriquement dans la scène, sous la forme de maillages triangulés. Mais pour le rendu temps réel, utiliser une telle représentation pour l'ensemble des micro-objets tels que ceux listés ci-dessus pose trois problématiques majeures :

- Une quantité de mémoire supérieure à celle disponible sur la carte graphique peut être requise pour leur stockage.
- La quantité d'objets peut dépasser la capacité de traitements géométriques simultanés de la carte graphique.
- La densité d'éléments couvrant un même pixel peut générer de nombreux artefacts de calcul (tels que de l'aliassage ou des scintillements lors de l'animation).

Pour ces éléments de surface, lorsqu'ils sont assimilables à des petits détails dans l'image, des représentations alternatives sont préférables afin de réduire la complexité du résultat au prix d'une perte de qualité contrôlée.

FIGURE 1.1 – Un champ d'herbe par Boulanger *et al.* [BPB09]

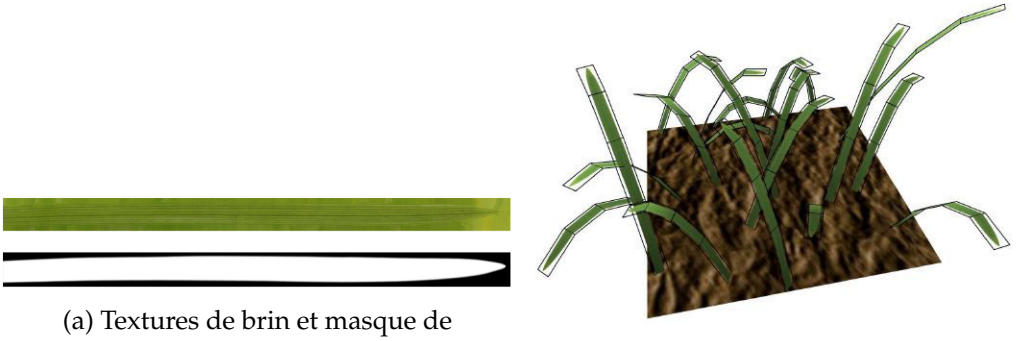
Il existe dans la littérature de nombreuses représentations possibles des objets, autres que les maillages de triangles. Pour le rendu de champ d'herbe en temps réel, la végétation de surface peut être modélisée par un ensemble de petites surfaces, par une image plaquée sur le terrain, ou encore par un volume de végétation au dessus sur le sol. Nous distinguons similairement les différentes représentations selon trois catégories correspondantes aux informations modélisées : les représentations surfaciques, les représentations basées image et les représentations volumiques.

1.1.1 Les représentations surfaciques

Les représentations surfaciques se focalisent sur la reproduction de la surface des objets, c'est-à-dire la jonction entre le volume de l'objet et le milieu contenant cet objet. Ces représentations peuvent être stockées en mémoire sous la forme d'un ensemble de primitives géométriques telles que des triangles, ou bien sous la forme de textures de relief décrivant la déformation d'une surface sous-jacente.

Les primitives géométriques

Les triangles, les quadrilatères ou encore les lignes anti-aliasées sont couramment considérés en premier lieu pour modéliser un objet : la visualisation de ces primitives peut être réalisée très rapidement par projection de celles-ci sur un plan représentant l'écran et par rastérisation des primitives projetées. Ce processus de transformation de données vectorielles 3D en images 2D est implanté matériellement dans les cartes graphiques. Le nombre de primitives pouvant être traitées en parallèle en un temps minimal ne cesse d'augmenter avec l'évolution de ces cartes, permettant de rendre des scènes de plus en plus complexes en temps réel.



(a) Textures de brin et masque de transparence

(b) Géométrie texturée des brins

FIGURE 1.2 – Représentation géométrique des brins par Boulanger *et al.* [BPB09]
Chaque brin est constitué d'une chaîne de quadrilatères sur laquelle est plaquée une texture associée à un masque de transparence.

Les lignes anti-aliasées sont parmi les premières primitives à avoir été proposées dans la littérature pour le stockage et le rendu de brins [RB85]. Un ensemble de points 3D décrivant les différents segments, associés à différents coefficients d'épaisseur, permet de rapidement obtenir à moindre coût des objets filiformes simples rendus par la

rastérisation directe des différents segments. Mais cette représentation est limitée à une forme très particulière. Or l'apparence de nombreuses espèces d'herbe est souvent plus proche d'une lame courbe que d'un ensemble de segments. S'il est possible de représenter cette lame par un ensemble de triangles [FLHS15], une approche courante est l'utilisation d'une chaîne de quadrilatères texturée [BPB09, QCCL12, Pap15]. Cette dernière est plus simple à concevoir géométriquement, mais requiert une gestion efficace de la transparence pour limiter les artefacts d'aliassage et de transparence. Dans la pratique, pour limiter le stockage important induit par ce grand nombre de brins positionnés sur la surface, un nombre réduit de patchs de brins géométriques est conservé. Ces patchs sont redistribués sur une grille régulière lors du rendu pour générer le champ (voir section 1.2.1). Boulanger *et al.* [BPB09] utilisent de tels patchs (figure 1.2) comme représentation du champ proche caméra, mais également comme modèles pour la génération de représentations alternatives, décrites dans les parties suivantes.

Les courbes paramétriques

Une courbure lisse est importante pour le réalisme du résultat, en particulier lorsque les brins sont vus de très près. L'utilisation de chaînes de quadrilatères ou de segments de lignes permet d'approximer cette courbure, mais une discrétisation importante de la géométrie peut être nécessaire pour obtenir une courbure lisse, augmentant le coût mémoire et le temps de rendu. Une possibilité offerte par les capacités grandissantes des cartes graphiques est l'utilisation de représentations exprimant cette courbure analytiquement dans leur formulation. Les courbes paramétriques permettent cela, en plus d'être plus légères en mémoire comparativement à un ensemble de polygones, au prix d'une évaluation moins optimisée par le pipeline graphique.

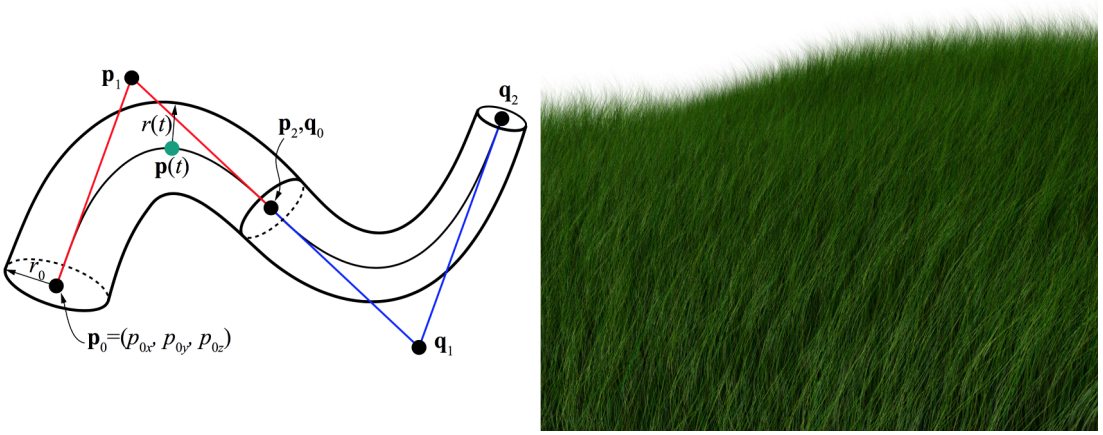


FIGURE 1.3 – Rendu d'herbe par Barringer *et al.* [BGAM12]
Chaque brin est constitué d'une ou plusieurs courbes de Bézier.

Les courbes paramétriques sont souvent utilisées pour représenter des éléments très fins tels que des cheveux et de la fourrure, les plus couramment utilisées pour la fourrure et l'herbe étant les courbes de Hermite [AMH08] et de Bézier [BGAM12]. Une courbe de Hermite est caractérisée par deux points de contrôle (p_0, p_1) et des tangentes (m_0, m_1) de la courbe à ces deux points. Une position sur la courbe générée à partir de ces informations peut être obtenue par interpolation :

$$p(t) = p_0(2t^3 - 3t^2 + 1) + m_0(t^3 - 2t^2 + t) + p_1(-2t^3 + 3t^2) + m_1(t^3 - t^2) \qquad t \in [0, 1] \quad (1.1)$$

Similairement, un point sur une courbe de Bézier caractérisée par quatre points de contrôle (p_0, p_1, p_2, p_3) peut être obtenu par l'interpolation suivante :

$$p(t) = p_0(1-t)^3 + 3p_1t(1-t)^2 + 3p_2t^2(1-t) + p_3t^3 \qquad t \in [0, 1] \quad (1.2)$$

Un des principaux avantages de la courbe de Bézier est sa possibilité d'être reconstruite géométriquement par une subdivision récursive, mais les courbes de Hermite permettent un contrôle plus aisé de la forme pour la modélisation.

Ces représentations peuvent être transformées par le pipeline graphique en une géométrie discrète adaptée au niveau de détails désiré (voir section 1.2.2), ou directement évaluées par pixel. Fan *et al.* [FLHS15] utilise la première de ces deux solutions : différents modèles de brins sont représentés par des profils de courbes puis transformés en géométrie lors du rendu. Barringer *et al.* [BGAM12] modélisent de la fourrure et de l'herbe en utilisant des courbes de Bézier (figure 1.3), mais ils proposent également un pipeline de rendu pour évaluer directement l'apparence de ces courbes.

Plaquage de relief

Une autre approche possible consiste à représenter les détails de surface non comme des objets à part entière, mais comme une déformation du relief de la surface sous-jacente. Ces déformations sont stockées en mémoire sous la forme d'une ou plusieurs textures décrivant la modification de la surface. Pour faire le rendu de ces déformations, ces textures sont associées à différentes méthodes de génération ou de simulation de relief, telles que le plaquage de normales ou le plaquage de déformations.

Kuhnert *et al.* [KB12] utilisent cette idée de déformation de la surface pour simuler une apparence de fourrure : différentes textures (figure 1.4.a) sont utilisées pour paramétrer un algorithme de *cone step mapping*. Cette méthode se base sur l'avancée pas à pas du rayon par test d'intersection avec un cône englobant l'espace libre minimal pouvant être parcouru par le rayon. Ce cône est précalculé pour chaque unité de texture du champ de hauteur. La version proposée par Kuhnert *et al.* utilise également une carte de déformation pour modifier le parcours du rayon et créer l'effet de fourrure présenté dans la figure 1.4. Mais cette méthode, tout comme les méthodes de plaquage de déformations ou de normales, ne modifie pas la géométrie de l'objet : la silhouette de l'objet dans l'image reste identique sur les angles rasants.

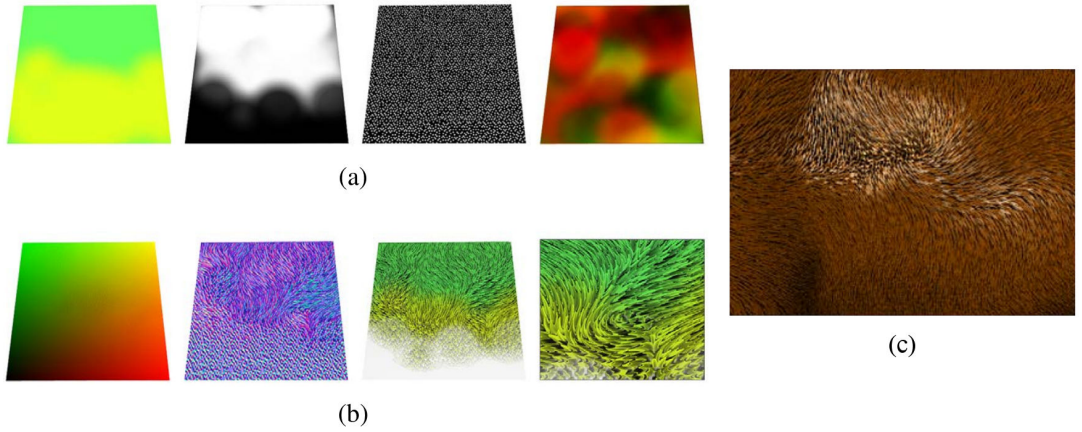


FIGURE 1.4 – Fourrure représentée par déformation de relief et rendu par *Cone Step Mapping* (CSM) [KB12]. a) De gauche à droite : les textures de couleur, de longueur des brins, de hauteur et de distorsion utilisées par l'algorithme de CSM de Kuhnert *et al.* b) De gauche à droite : l'espace texture modifié, les normales générées, le résultat et un zoom sur le résultat obtenu par leur méthode. c) Une fourrure obtenue par cette méthode. Aucune géométrie n'est générée en supplément de celle de l'objet sur lequel est appliqué la méthode, y compris sur les bords de l'objet dans l'image.

1.1.2 Les représentations basées image

Les représentations basées image s'abstraient de la géométrie et même du volume d'un objet pour ne considérer que l'apparence produite par celui-ci selon différents points de vue. Ces représentations diminuent la complexité géométrique d'un objet par l'utilisation de photographies ou de textures prérendues des objets. Ces représentations sont depuis longtemps utilisées dans les jeux vidéo pour donner l'illusion de la végétation à moindre coût. Pour représenter de la végétation, des images représentant le visuel attendu peuvent être directement plaquées sur la surface du terrain ou sur des imposteurs géométriques, ou encore être utilisées pour créer une *Fonction de Texture Bidirectionnelle*.

Plaquage de texture

Pour modéliser une apparence complexe à moindre coût, une texture plaquée sur le sol de la scène est souvent préférée dans les jeux vidéo cherchant avant tout la performance. Mais de telles textures peuvent aussi être utilisées comme représentation alternative des détails de surface sur les zones éloignées de l'observateur pour le rendu quasi réaliste. Cette texture peut être suffisante pour représenter des détails supposés trop petits dans l'image pour être perçus comme des éléments géométriques.

Des textures produites par prérendu de végétation géométrique vue de dessus [BPB09, QCCL12] ou par des méthodes procédurales [Pan14] peuvent être plaquées directement sur la surface, en particulier pour les zones du sol loin de l'observateur afin de produire à minima l'apparence d'une végétation de surface. Ces textures peuvent également être combinées à d'autres représentations pour une visualisation plus proche de la caméra : à moyenne distance de vue, associées à une représentation géométrique simplifiée, ces textures peuvent combler les zones où la densité de géométrie est réduite [PC01].

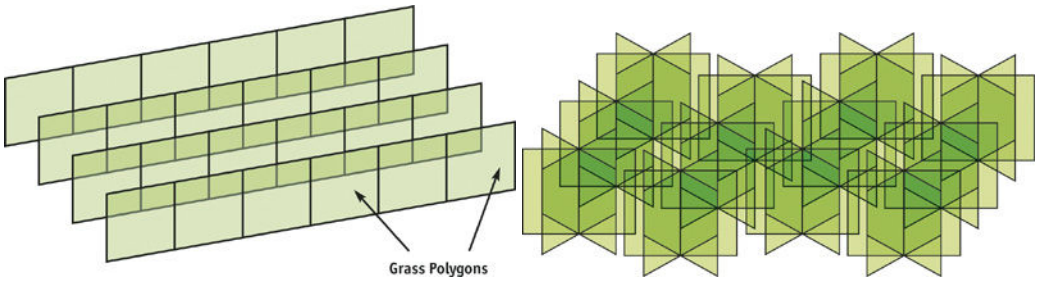
Les imposteurs

Une ou plusieurs textures représentant le visuel d'un objet sous plusieurs angles de vue peuvent également être utilisées pour remplacer l'objet géométrique complexe par de simples quadrilatères sur lesquels ces différentes textures sont plaquées. Cette représentation conserve une forme de silhouette géométrique mais approxime, voire sacrifie, les informations de parallaxe. Cette approximation peut être pertinente pour représenter un objet dont la forme géométrique générale peut être distinguée dans l'image, mais dont les détails de surface restent difficilement identifiables ou varient peu en fonction de la position de l'observateur autour de l'objet.

Ces quadrilatères texturés sont plus couramment appelés *imposteurs* ou *billboards*. Les textures appliquées sur ces derniers peuvent être créées par photographies, par rendu dans des textures d'un objet selon différents points de vue, ou encore par des méthodes procédurales. Les premiers jeux en 3D utilisaient dans de nombreux cas un unique quadrilatère sur lequel était plaquée une texture animée et faisant constamment face à la caméra. Mais le mouvement de caméra laisse clairement percevoir à l'utilisateur le plan sur lequel est plaquée cette texture lorsque l'observateur se rapproche de celui-ci.

Les imposteurs sont souvent utilisés pour représenter des éléments de végétation tels que des buissons, le feuillage des arbres ou des touffes d'herbe à courte et moyenne distance de vue [BCF*05, QCCL12]. Pour représenter de larges champs d'herbe, des imposteurs constitués de chaînes verticales de quadrilatères peuvent être utilisés [PC01]. Des textures formant des bandes de brins d'herbe sont plaquées sur ces chaînes, les bandes formées par ces dernières étant alignées selon deux directions. La bande de quadrilatères à afficher dans ce cas peut être choisie en fonction de l'orientation de la caméra afin de masquer les bandes parallèles à cette orientation. La chaîne de quadrilatères permet la mise en place d'une animation simple par rotation des différents quadrilatères pour créer une courbure animée. L'utilisation d'imposteurs sous la forme de simples quadrilatères croisés en étoiles ou en triangles [Pel04] minimise les problèmes de parallaxe, et par extension les risques que l'observateur perçoive la véritable géométrie utilisée (figure 1.5). Depuis l'apparition des *fragment shaders* dans le pipeline graphique et la programmation des accès textures par pixel, il est possible d'utiliser des imposteurs faisant face à la caméra tout en approximant la parallaxe. La texture à plaquer sur l'imposteur peut être sélectionnée dans le *fragment shader* en foncton de la direction de vue pendant le rendu [Wha05]. La parallaxe est approximée pour une direction donnée par interpolation des textures correspondant aux directions sauvegardées les plus proches.

Dans chacun de ces cas, une contrainte de perpendicularité au sol est imposée aux imposteurs de végétation. Cet alignement est essentiel pour ne pas obtenir des brins suivant strictement le comportement du plan caméra.



(a) Imposteurs alignés (b) Imposteurs croisés

(c) Champ d'herbe utilisant des imposteurs croisés

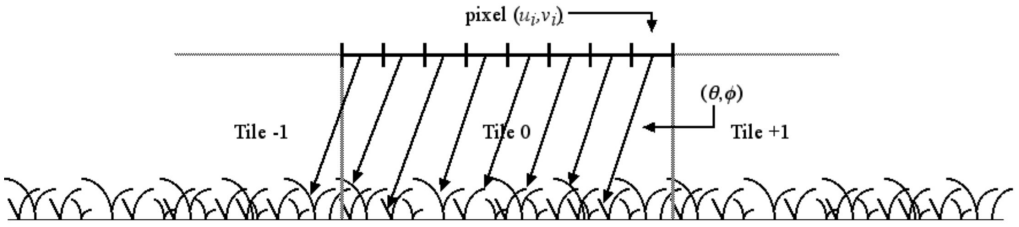
FIGURE 1.5 – Imposteurs géométriques par Pelzer [Pel04]
Des textures représentant un ensemble de brins sont appliquées sur ces imposteurs géométriques pour simuler une végétation plus dense.

Fonctions de textures bidirectionnelles

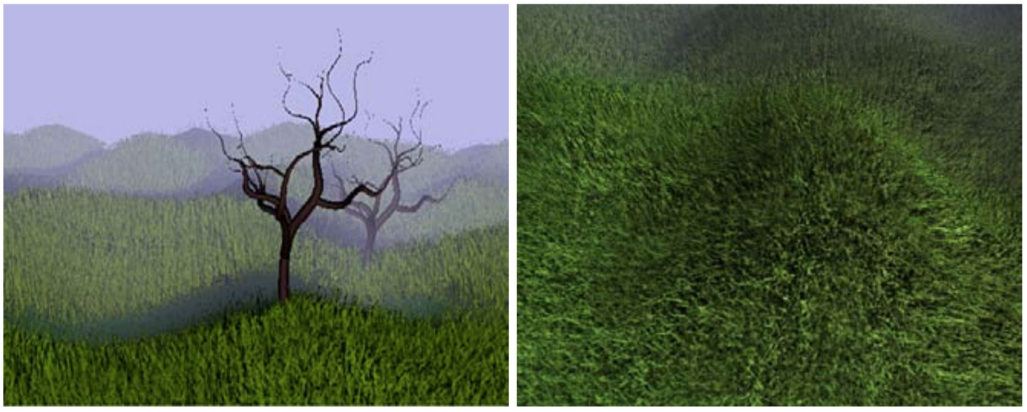
Les fonctions de textures bidirectionnelles (ou **BTFs** pour *Bidirectional Texture Functions*) [DvGNK99] sont une simplification des *Bidirectionnal Surface Scattering Reflectance Distribution Functions* : une **BTF** est une fonction 6-dimensions prenant en entrée un point (x) d'une surface S, une direction de rayon de vue (θ_i, ρ_i) et une direction de rayon de lumière (θ_r, ρ_r), et renvoyant pour ces données une luminance approximant le comportement visuel du matériau de surface en ce point [FP07].

$$BTF_{rgb}(x, \theta_i, \rho_i, \theta_r, \rho_r) = \int_S BSSRDF(x_i, x_r, \theta_i, \rho_i, \theta_r, \rho_r) dx_i \tag{1.3}$$

Une **BTF** peut être créée par la photographie, le prérendu d'un objet ou d'autres méthodes d'acquisition [Fil09] selon ces différents angles de vue et directions de la lumière. Pour le rendu temps réel d'objets de végétation, des **BTFs** représentant des objets complexes peuvent être créées sous la forme d'imposteurs : un objet est transformé en un ensemble de textures par prérendu de celui-ci pour chaque direction de vue et de lumière souhaitée [MNP01]. Lors du rendu, cet objet peut être remplacé par un ensemble d'imposteurs quasi perpendiculaires à la caméra, ou par un unique imposteur sur lequel une



(a) Acquisition de **BTF**



(b) Rendu d'une scène d'herbe

FIGURE 1.6 – Acquisition et rendu de **BTF** par Shah *et al.* [SKP05]
La **BTF** est construite par rendu d'un groupe de patchs géométriques selon différents angles de vue. Une information de profondeur est également conservée pour effectuer des tests d'occultation et de transparence.

texture sera calculée par interpolation des textures correspondant aux directions de vue et de lumières les plus proches. Une telle **BTF** peut être également utilisée pour représenter de l'herbe (ou une quelconque végétation) sous la forme de patchs de surface [SKP05] (figure 1.6).

Si la complexité géométrique est grandement réduite tout en gardant un résultat visuel de qualité, le nombre de textures nécessaires à la création d'une **BTF** haute résolution demande une grande quantité de mémoire pour le stockage de celle-ci. La diminution du temps de calcul liée à la complexité réduite est dans ce cas contrebalancée par une utilisation plus intensive de la mémoire lors du rendu, autant en quantité de mémoire requise qu'en nombre d'accès à celle-ci. Pour réduire la taille en mémoire et/ou augmenter la résolution de ces textures, il est possible d'utiliser des algorithmes de compression [SKP05], mais la taille et le nombre d'accès mémoires restent prohibitifs pour le rendu temps réel. Le cas idéal pour l'utilisation des **BTFs** se situe à moyenne et longue distance de l'observateur, lorsque le besoin de précision des détails est moindre et qu'une **BTF** basse définition peut être suffisante.

1.1.3 Les représentations volumiques

Contrairement aux représentations surfaciques ou basées image, les représentations volumiques modélisent un objet comme un volume défini dans un espace 3D. Pour re-

présenter un objet, ce volume est discrétisé en un ensemble de points élémentaires ou en un ensemble de textures 2D représentant des tranches de l'objet. Ces représentations volumiques peuvent être utilisées pour modéliser un volume de détails au-dessus d'une surface en distribuant des objets volumiques sur celle-ci. Pour limiter le coût de stockage et de rendu, ces détails peuvent être plus spécifiquement considérés comme une surcouche de matière plaquée sur la surface, créée par empilement de tranches parallèles à la surface.

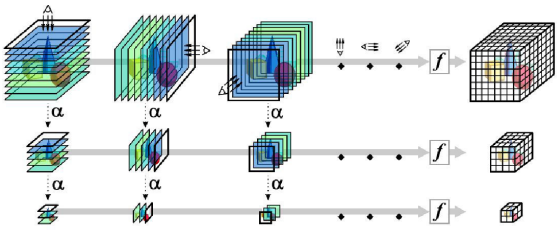
Objets et textures volumiques

Pour des objets complexes, une alternative possible à la géométrie est de modéliser les objets sous la forme de volumes discrets définis par un nuage de points associés à une matière particulière. Ces éléments de volume, ou *voxel*, peuvent être distribués de manière éparse, irrégulière ou bien régulière dans un espace 3D. Dans ce dernier cas, la grille de voxels définissant un objet peut être stockée sous la forme d'une texture 3D en mémoire, ou dans une structure accélératrice telle qu'un octree [LHN05] pour optimiser son occupation mémoire et son évaluation. Ce type de représentation est couramment utilisé pour la visualisation médicale : elle permet de modéliser de manière plus réaliste des objets semi-transparents pouvant être composés de plusieurs matériaux. Une méthode de rendu basée sur une stratégie d'échantillonnage adaptée est cependant nécessaire pour l'évaluation du transport de la lumière dans ce volume (voir section 1.4).

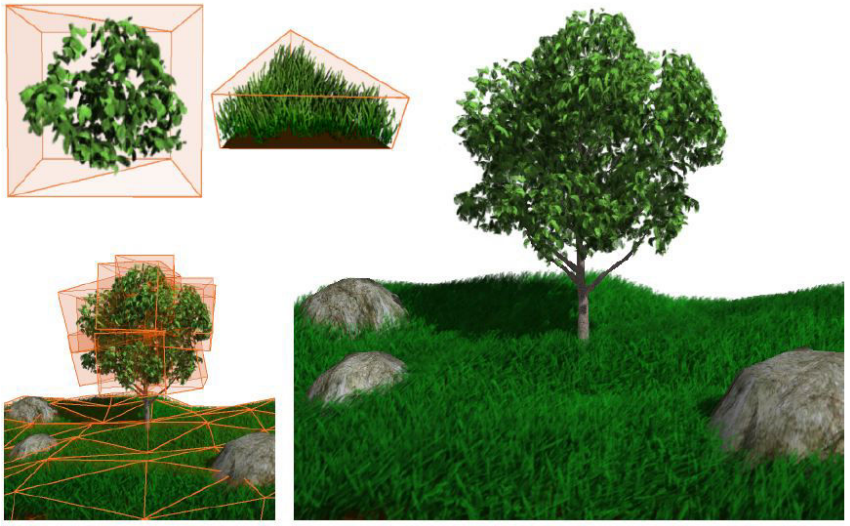
Pour modéliser les détails d'une surface tels que de la fourrure, une texture volumétrique de surface peut être créée par la distribution de volumes de référence sur une surface géométrique [KK89]. Pour profiter de l'accélération des processeurs graphiques ne gérant pas nativement les textures 3D, ces volumes peuvent être créés sous la forme d'un ensemble de textures 2D obtenues par prérendu de coupes de l'objet [MN98,BPB09]. Un découpage du volume selon trois directions orthogonales permet de limiter l'apparition de vides lors de l'évaluation. Dans le but d'améliorer les représentations basées sur les imposteurs, Decaudin et Neyret [DN09] créent des textures volumiques à l'aide d'imposteurs volumiques : des prismes associés à des textures 3D sont utilisés pour tuiler des surfaces (figure 1.7). Ces textures 3D sont cette fois générées par rendu de coupes selon six directions de vue, alignées selon les axes d'un cube englobant. Des *MIP-maps* [MSR16] peuvent être générées à partir de ces textures pour obtenir un rendu visuellement correct de l'imposteur à différents niveaux d'échelle dans l'image. Des algorithmes de compression peuvent être appliqués sur ces *MIP-maps* pour en réduire l'empreinte mémoire.

Plaquage de surcouches

Définir les détails de surface comme un ensemble d'objets volumiques requiert une grande quantité de mémoire pour le stockage des différents objets, et des méthodes de rendu complexes pour le rendu temps réel. Cette complexité est notamment liée aux différentes stratégies d'échantillonnage des différents volumes. Ces détails peuvent également être considérés comme une surcouche de matière plaquée sur un objet [CTW*04, PBFJ05].Chen *et al.* [CTW*04] définissent en ce sens une fonction de texture de surcouche



(a) Création de la texture volumique



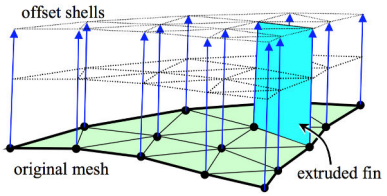
(b) Champ d'herbe rendu en utilisant des imposteurs volumiques

FIGURE 1.7 – Les imposteurs volumiques par Decaudin et Neyret [DN09]
(a) La texture volumique et ses *MIP-maps* sont obtenues par rendu selon différentes directions d'un ensemble d'objets. (b) Des tétraèdres sont ensuite utilisés comme géométrie de support pour le rendu de la texture.

comme un matériau hétérogène composé de strates de voxels pour modéliser une méso-structure volumique plaquée sur la surface. Le transport de la lumière dans ce matériau est évalué par lancer de rayon dans la surcouche. Cette fonction de texture, définie par un ensemble restreint de voxels, limite la quantité de mémoire nécessaire pour la modélisation de détails de surface, mais le rendu par échantillonnage du rayon utilisé par Chen *et al.* demande un temps d'évaluation important. D'autres formes de plaquage de surcouche permettent une évaluation rapide par une carte graphique et sont construites sur un modèle commun : une géométrie de support est générée par des extrusions successives du maillage de la surface, sur lesquelles sont plaquées différentes textures 2D.

Pour créer de la fourrure ou de la végétation de surface, une approche très similaire à celle proposée par Meyer et Neyret [MN98] peut être utilisée pour créer une surcouche volumique, mais en utilisant une unique direction de coupe : des textures de surcouche peuvent être créées par rendu en coupe d'un volume englobant un ensemble de brins géométriques [LPF01, BW06, GY13] (figure 1.8). Les coupes réalisées sont parallèles à la surface de référence, sur laquelle ces poils sont distribués, pour créer les différents niveaux de la surcouche. Pour camoufler les zones de vide visibles sur les angles rasants de la surcouche obtenue, des textures de silhouette peuvent être générées par prérendu de cette

(a) Textures de surcouche et de bords



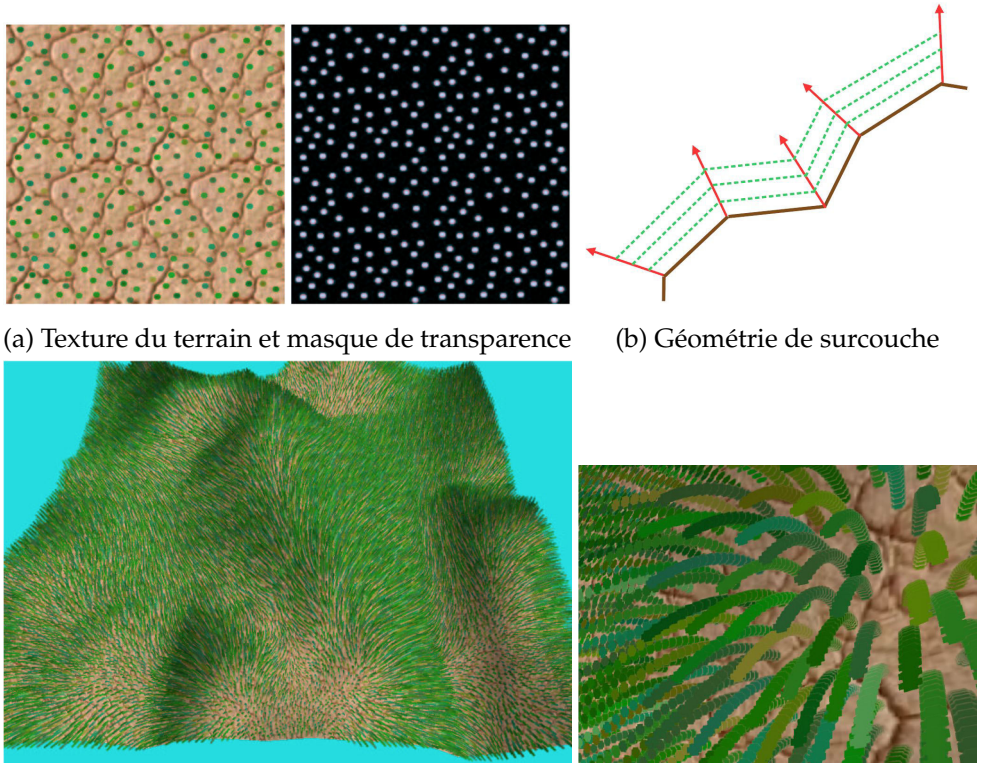
(b) Géométrie de surcouche pour le rendu

(c) Surcouche de fourrure appliquée sur le Stanford Bunny

FIGURE 1.8 – Surcouche de fourrure par Lengyel *et al.* [LPF01]
Plusieurs textures 2D (a) sont appliqués sur différents niveaux d'extrusion du maillage et sur des plans perpendiculaires à la surface (b).

même géométrie de référence. Ces textures de silhouette requièrent dans ce cas qu'une géométrie de support perpendiculaire à la surface soit générée. Une représentation très simplifiée de ces textures de surcouche est proposée par Bakay et Heidrich [BH02] (figure 1.9). Une unique texture de terrain est utilisée pour stocker la couleur du résultat, cette texture comprenant l'apparence de la surface (c'est-à-dire une texture de sol) et des points colorés représentant la position et la couleur des brins. Cette texture est associée à un masque de transparence isolant la position des brins et à une texture de hauteur dans laquelle est stockée la taille des différents brins d'herbe. Lors du rendu, la texture du terrain est plaquée sur l'ensemble des couches, mais chaque couche subit également un test de transparence lié à son masque et à la carte de hauteur des brins.

À plus grande échelle, les textures de surcouche peuvent être utilisées pour représenter des objets plus complexes. Behrendt *et al.* [BCF*05] utilisent par exemple des textures de surcouche pour le rendu de végétation située à une distance lointaine, à partir de laquelle les objets sont trop petits à l'écran pour nécessiter l'utilisation de géométrie ou d'imposteurs visuels. Pour une comparaison plus détaillée des différentes représentations et méthodes de rendu liées au plaquage de surcouche, nous invitons le lecteur à consulter l'état de l'art de Koniaris *et al.* sur le plaquage et le rendu de mésostructures volumiques [KCYM14].



(a) Texture du terrain et masque de transparence (b) Géométrie de surcouche

(c) Surcouche d'herbe appliquée sur un terrain (d) Vue de près

FIGURE 1.9 – Surcouche d'herbe par Bakay et Heidrich [BH02]
La texture de sol ((a) gauche) est appliquée sur différents niveaux d'extrusion du maillage (b) avec un test de transparence selon un masque ((a) droite). Le résultat est convaincant de loin mais l'espacement entre les couches est visible de près (d).

1.1.4 Synthèse

Toutes ces représentations ont leurs points forts et leurs points faibles, synthétisés ci-après dans la figure 1.10. Pour la modélisation de détails en particulier, ces représentations ont toutes une utilisation optimale possible en fonction de la taille des détails projetés dans l'image et de leur impact visuel. Partant de ce constat, leur utilisation dans une stratégie de gestion du niveau de détails est préférée dans de nombreuses implantations [BPB09, QCCL12].

Les représentations basées sur les primitives géométriques ne montrent leur plein potentiel que lorsque ces primitives recouvrent un certain nombre de pixels. Dans le cas des champs d'herbe lointains, le nombre de primitives implique un surcoût en complexité et en difficulté de filtrage. Les détails semi-transparents modélisés par de la géométrie requièrent de plus une stratégie de mélange optimisée pour obtenir un visuel correct. Pour limiter la complexité géométrique, les concepteurs préfèrent utiliser d'autres représentations moins précises des micro-objets de surface, mais suffisantes pour faire illusion. Le plaquage de relief en particulier permet de limiter grandement la complexité en utilisant des méthodes de simulation de relief adaptée. Le préfiltrage des informations de relief [BN11] peut également permettre à ces méthodes de générer efficacement un résultat

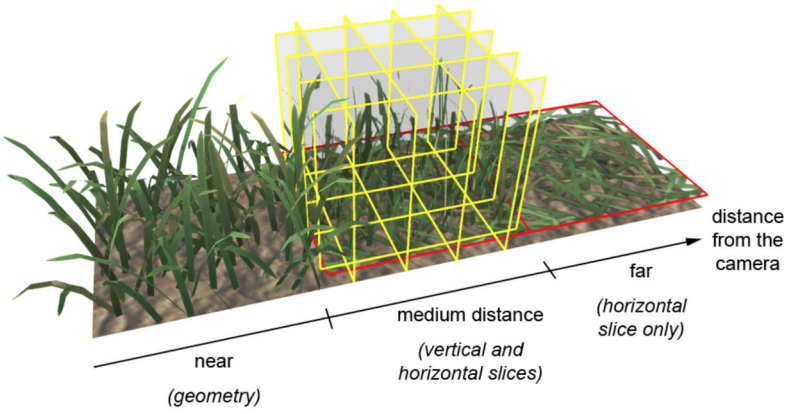
Représentations	Qualité	Vitesse	Mémoire
Surfaciques :			
- Primitives	+	+	~
- Courbes	+	−	+
- Relief	+	+	+
Images :			
- Textures	−−	++	+
- Imposteurs	−	++	~
- BTF	++	−−	−−
Volumiques :			
- Objets volumiques	++	−	−−
- Textures de surcouche	+	++	~

FIGURE 1.10 – Synthèse des points forts et points faibles des représentations de champs d'herbe et de fourrures.

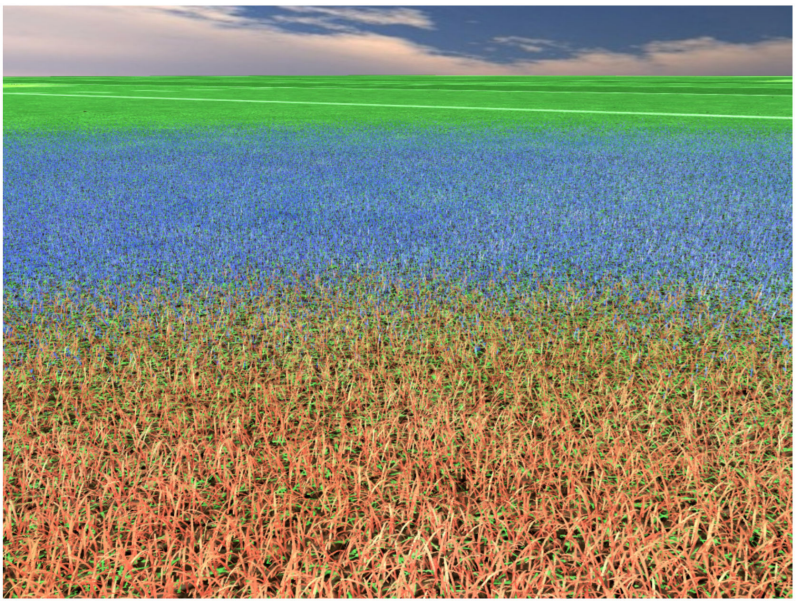
anti-aliasé. Les méthodes de simulation de relief se focalisent cependant sur la simulation de l'apparence d'une géométrie de la surface sans modifier les contours dans l'image de cette même géométrie.

Les représentations basées image permettent une diminution maximale de la complexité géométrique. Mais cette transformation du détail 3D en 2D supprime de nombreuses informations nécessaires au rendu réaliste, limitant ce réalisme à des positions particulières de l'observateur et de la lumière. La faible complexité de cette représentation en fait cependant un très bon choix pour modéliser des détails de surfaces très lointains, lorsque la géométrie des détails ne peut être distinguée de celle de la surface. Les imposteurs apportent une solution à ce manque de géométrie, mais le manque de parallaxe limite également le réalisme des objets lorsque l'observateur se déplace près de ces derniers. Des problèmes de complexité géométrique, de filtrage et de gestion de la transparence persistent également pour le rendu de très nombreux imposteurs semi-transparents.

Les représentations volumiques présentent un potentiel grandissant pour le rendu temps réel. L'évolution des cartes graphiques permet d'implanter des méthodes de rendu volumétriques de plus en plus complexes pour la production d'effets volumiques naturels toujours plus réalistes (voir section 1.4). Pour le rendu temps réel, ces représentations peuvent utiliser des textures de données accessibles de manière arbitraire par les cartes graphiques. Certaines de ces textures sont compatibles avec les méthodes de filtrage des représentations basées image telles que les *MIP-maps*. Des structures accélératrices telles que les *octree* ont été implantées sur cartes graphiques avec succès, et permettent également d'optimiser la complexité ou la qualité des objets rendus en fonction de la distance de l'objet à l'observateur. Malgré ces atouts, ces représentations présentent toujours un problème sensible pour le rendu temps réel de scènes très détaillées : pour la modélisation de détails tels que des brins, une discrétisation importante du volume peut être nécessaire pour une reproduction de qualité suffisante du résultat.



(a) Représentations selon la distance de vue



(b) Répartition des représentations dans une scène

FIGURE 1.11 – Gestion du niveau de détails par Boulanger *et al.* [BPB09]
(a) Trois représentations différentes sont utilisées pour adapter la complexité du rendu en fonction de la distance de l'observateur à la surface. (b) Les brins géométriques sont ici rendus en rouge, ceux de la texture volumique sont rendu en bleu, et ceux de la texture de sol sont rendus en vert.

À travers cette thèse, nous étudions la possibilité de créer des détails similaire à des brins sous la forme d'une texture de surcouche volumique procédurale, construite comme une distribution de volumes. Une telle texture permet une modélisation volumique continue des détails pour un coût minime en mémoire, cette texture étant stockée sous la forme d'un programme d'évaluation par point de l'espace. Nous considérons deux possibles représentations de support pour l'évaluation de cette texture : une approche similaire au plaquage de surcouche de Chen *et al.* [CTW*04] peut être utilisée, considérant notre texture procédurale comme une texture de surcouche définissant un matériau hétérogène. Mais, de par sa construction, cette texture peut être également utilisée dans une approche similaire à celle de Decaudin et Neyret [DN09] ou de Boulanger *et al.* [BPB09], à savoir l'évaluation de patchs volumiques distribués sur la surface.

1.2 La génération de détails surfaciques à la volée

Créer plusieurs millions d'objets à la fois similaires et différents est un processus long et complexe du point de vue artistique. Pour accélérer et simplifier le processus de création, les logiciels de modélisation proposent aux artistes des méthodes procédurales capables de générer ces détails à partir de paramètres. Avec l'évolution du matériel graphique, les méthodes procédurales basées sur la tessellation dynamique ou celles évaluables par pixel peuvent intégrer les moteurs de rendu temps réel et permettent de générer ces détails lors du rendu. Ces méthodes permettent de diminuer drastiquement la taille mémoire de la scène : une surface détaillée, composée de millions de triangles ou sur laquelle est appliquée une texture très haute définition, peut être remplacée par une surface composée de quelques milliers de triangles sur laquelle des programmes génèrent les éléments de détails lors de l'affichage finale de la scène. Ces méthodes permettent également d'adapter la complexité de la scène lors du rendu en fonction de différents critères, tels que l'échelle des objets rendus à l'écran ou encore les performances du matériel.

Plusieurs approches permettent de générer les détails de surface à la volée lors du rendu. Une approche possible pour une génération rapide est l'utilisation d'une fonction de distribution, associée à un ensemble d'objets préexistants, pour paver l'espace en distribuant aléatoirement ces objets. L'utilisation d'objets explicites (stockés sous la forme de triangles, de volumes ou de textures) limite cependant le résultat à une résolution maximale définie par la résolution de ces objets. Ces fonctions de distribution peuvent être associées à des objets eux aussi procéduraux, générés ou évalués par la carte graphique, pour créer une modélisation entièrement procédurale évaluable en temps réel. Ces objets en particulier peuvent être de la géométrie obtenue par une subdivision contrôlée de la surface, ou bien encore des primitives paramétriques évaluables par point et décrivant un volume ou une texture.

Nous récapitulons ci-après les différentes méthodes proposées dans la littérature pour le rajout de détails à la volée pour le rendu temps réel. Nous détaillons les différentes approches de distribution interactive d'objets, suivies des deux grandes familles de méthodes utilisées pour la génération procédurale de détails en temps réel : la subdivision de surface pour la génération de géométrie, et les textures procédurales pour la génération de textures surfaciques ou volumiques. Nous nous concentrons principalement sur les méthodes permettant de générer des champs d'herbe ou de la fourrure en exploitant la carte graphique. Pour une liste plus exhaustive des méthodes de génération procédurale des différents contenus d'une scène, nous invitons le lecteur à consulter l'état de l'art de Hendrikx *et al.* [HMVI11].

1.2.1 La distribution procédurale de détails

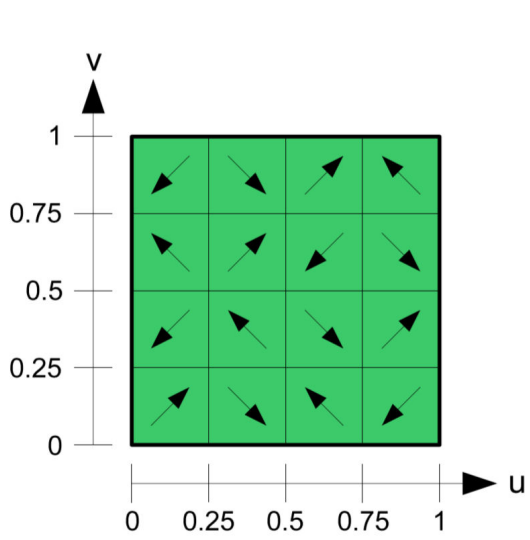
La distribution d'objets, ou d'instances d'objets, directement lors du rendu permet de grandement limiter la géométrie stockée dans la scène. L'ensemble des détails peut être réduit à quelques modèles répétés sur la surface, cette répétition étant camouflée par une position et une orientation aléatoires de l'objet distribué. Les processus de distributions se basent sur des *générateurs de nombres pseudo-aléatoires* (ou **PRNG**), tels ceux présentés par Salmon *et al.* [SMDS11]. Une distribution pour laquelle la densité d'éléments ou les représentations d'objets utilisées peuvent être modifiées interactivement par zone de surface permet de gérer différents niveaux de détails et d'optimiser la complexité de la scène rendue. La distance de l'observateur à une zone ou la puissance du matériel utilisé peuvent par exemple servir de critères de sélection de la densité d'éléments à distribuer.

Le tuilage de surface

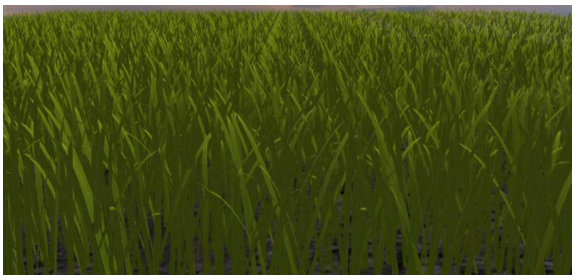
Pour faciliter le pavage d'une surface ou d'un volume par des objets, il est courant de découper cette surface en tuiles carrées [CSHD03] ou en polygones triangulés [PFH00], et par extension ce volume en cubes [LEQ*07] ou en prismes [LPF01]. Chaque tuile est ensuite associée à un patch élémentaire choisi au hasard lors du rendu. Cette approche n'est cependant pas sans défaut si l'on veut créer des motifs non répétitifs : si trop peu de patchs différents sont disponibles, leur répétition devient visible. Une coupe franche peut également apparaître entre les différents patchs si les couleurs de leurs bords ne correspondent pas. Les *Wang Tiles* et *Wang Cubes* [CSHD03, LEQ*07] minimisent ce problème en définissant des couleurs pour chaque arête (ou face) de la grille de tuiles. Un patch peut être associé à une tuile si ses couleurs d'arêtes ou de faces correspondent à celles des la tuile évaluée. D'autres approches résolvent ce problème en recalculant une coupe idéale entre deux patchs s'intersectant [LM07] : chaque patch appliqué à une tuile dépasse légèrement sur les tuiles voisines, une couture idéale étant calculée au niveau de l'intersection entre les patchs de chaque tuile pour recréer une continuité. Une autre approche possible, utilisée par Lengyel *et al.* [LPF01] pour la fourrure, consiste à recalculer une paramétrisation de la surface, pour déterminer à quels endroits doivent être agencés les différents patchs [PFH00]. Pour être évaluable en un point précis, ces différents processus doivent réaliser un prétraitement de l'espace pavé pour précalculer les informations liés aux tuiles voisines à ce point.

Pour les patchs de fourrure ou de brins d'herbe, le problème de coupe est moins perceptible et peut être négligé, mais un effet de répétition peut persister. Il est possible de minimiser cet effet en utilisant un plus grand nombre de patchs et en sélectionnant aléatoirement le patch à appliquer pour chaque tuile. Il est également possible de compenser cette répétition avec un nombre de patchs réduit en générant une orientation aléatoire de patch pour chaque tuile, créant l'illusion d'une diversité [BPB09] (figure 1.12). Une autre approche possible est de recréer un patch à la volée lors du rendu de chaque tuile, en sélectionnant aléatoirement par tuile un sous-ensemble des brins du patch pour multiplier le nombre de patchs différents à l'écran [FLHS15].

Le tuilage de surface par des patchs de géométrie fixée permet de créer rapidement une grande densité de brins de surface. Mais la répétition des brins est inévitable malgré les techniques précédentes. Aucun nouveau brin n'est généré lors du rendu, les brins utilisés étant fixés par la représentation. Cette approche entraîne une limitation inhérente pour le contrôle local du résultat : hormis des modifications de la taille de la géométrie rendu, peu de variations des brins peuvent être incluses sans créer de nouveaux patchs ou de nouveaux éléments tels que des textures de brins alternatives ou de nouveaux modèles géométriques. Une solution possible est d'utiliser pour chaque brin une représentation paramétrique pour générer la forme et l'apparence de chaque brin lors du rendu et pouvoir modifier localement cette apparence via les paramètres de la représentation.



(a) Distribution d'orientations



(b) Tuilage simple

(c) Tuilage et orientation aléatoires

FIGURE 1.12 – Tuilage de surface en utilisant des patchs aléatoirement orientés [BPB09]
Le tuilage simple (b) laisse rapidement apparaître le fait qu'un unique patch de géométrie est répété sur la surface. En utilisant le même patch, les orientations aléatoires (a) permettent de créer une apparence complexe et de cacher cette répétition (c).

Distribution aléatoire à la volée

Dans les méthodes de tuilage de surface précédentes, la distribution des brins est précalculée hors rendu grâce à un système de particules [Ree83, BPB09, LPF01, FLHS15] puis stockée dans les différents patchs. Ce système de particules permet de produire un résultat en prenant en compte des contraintes spatiales telles qu'un espace minimal entre les brins, évitant les recouvrements de géométrie tels que deux brins s'entrecoupant. La distribution obtenue par ce système possède des caractéristiques très proches d'une distribution en disques de Poisson et d'un *bruit bleu* : cette distribution et ce bruit sont tous deux identifiables par l'existence d'une dépendance spatiale entre les points générés, de telle sorte qu'aucun autre point n'est distribué dans un cercle de rayon spécifique autour de chaque point. Cette propriété est importante pour la simulation de végétation, où chaque plante tend à empêcher la croissance d'autres végétaux dans son voisinage direct.

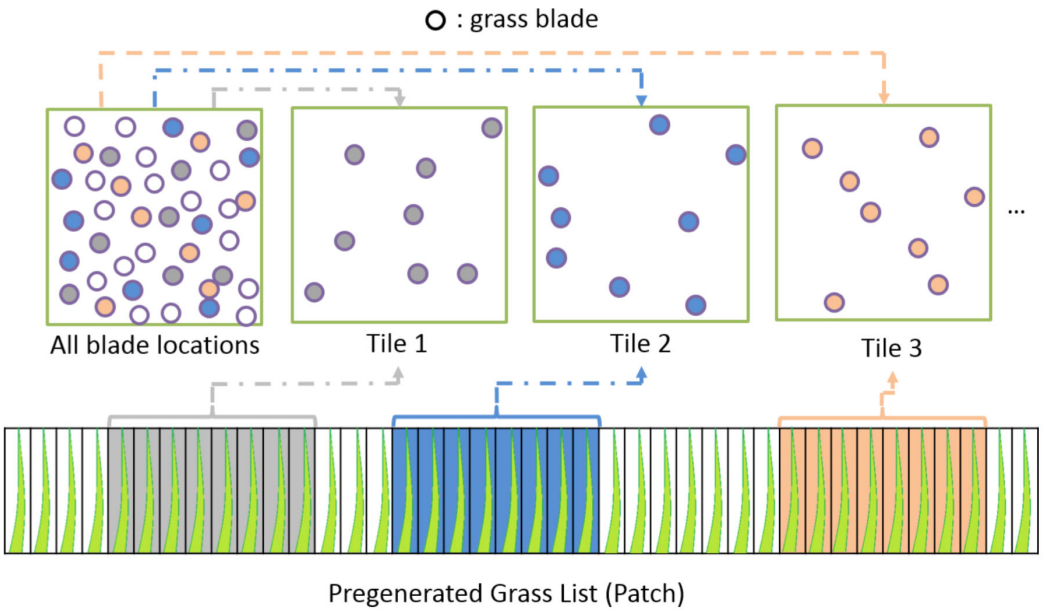


FIGURE 1.13 – Génération de patchs par Fan *et al.* [FLHS15]
À partir de la distribution de tous les brins et de leurs profils dans un patch élémentaire, le patch correspondant à une tuile de surface peut être généré par instanciation d'un sous-ensemble de brins.

Les premières méthodes de construction d'une distribution en disques de Poisson étaient réalisées par des itérations successives de génération aléatoire d'impulsions selon une distribution uniforme puis de rejet des impulsions se trouvant trop proches d'autres impulsions [Coo86]. Cette technique peut être réalisée en dérivant le pipeline graphique pour générer des impulsions aléatoires et rejeter les impulsions ne respectant pas le critère de distance minimale [IYLV13]. Ce processus appelé *dart throwing* doit être répété tant qu'un critère de couverture de la surface ou qu'un nombre d'itérations n'a pas été atteint, la génération d'impulsions en une unique itération ayant une faible probabilité de couvrir entièrement l'espace de distribution. Une autre approche possible consiste à utiliser un algorithme de relaxation [Llo82] déplaçant les impulsions générées au centre de leurs cellules de Voronoï respectives. Là encore, ce processus doit être itéré et de nouvelles impulsions peuvent être générées tant que des impulsions ne respectent pas le critère de distance minimale et que la couverture de l'espace est incomplète. Ces différentes approches peuvent créer une distribution à la volée mais un grand nombre d'itérations peut être nécessaire pour obtenir une distribution de Poisson de bonne qualité. Elles peuvent cependant être utilisées pour précalculer des patchs pouvant être utilisés dans une fonction de distribution [LD05, KCODL06] (figure 1.14).

Pour approximer cette distribution à moindre coût en une unique passe, une solution introduite par Cook [Coo86, Gla04] consiste à déformer une grille régulière d'échantillon par *jittering* : un déplacement aléatoire est ajouté à chaque échantillon d'une grille régulière (figure 1.15). Le *jittering* consiste plus généralement à ajouter une valeur de bruit à chaque échantillon d'un ensemble de données. Pour la création de champs d'herbe, le *jittering* de grille peut être utilisé pour calculer à la volée dans une tuile de surface les positions aléatoires des brins géométriques ou des imposteurs, en subdivisant la tuile en

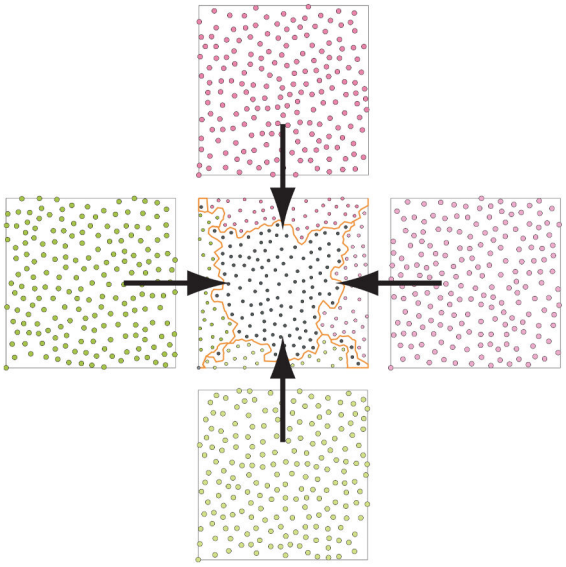
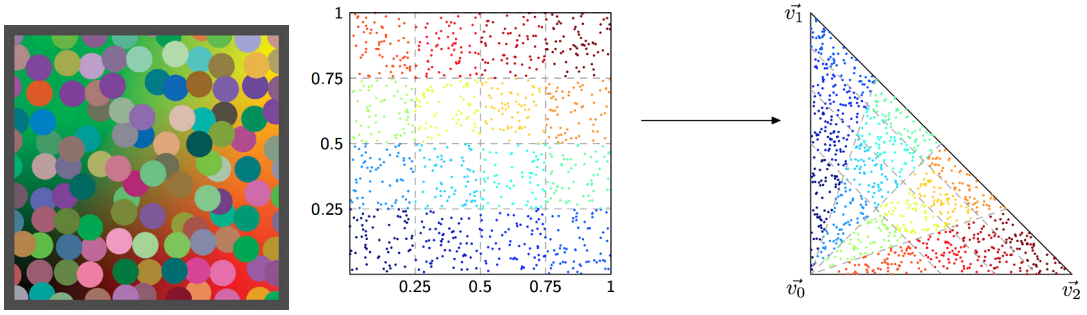


FIGURE 1.14 – Distribution de Poisson en utilisant des tuiles de Wang [KCODL06].
Pour chaque tuile, la distribution des points proches d'une arête est associée à une couleur. Cette couleur correspond à un sous-ensemble de tuile contenant une distribution de Poisson précalculée cohérente avec la distribution des points proches de cette arête.



(a) *Jittering* de grille [Gla04]

(b) *Jittering* multiple d'une grille régulière transposé dans un triangle [Pap15]

FIGURE 1.15 – Distribution de points par *Jittering*.

une grille régulière de points. Une approximation courante de ce *jittering* est de générer par cellule de la grille un ensemble uniforme d'impulsions aléatoires, ce qui revient à réaliser un *jittering* multiple de la grille initiale. Ce *jittering* peut également être appliqué à un maillage triangulé, en transposant la grille régulière d'échantillons aléatoires en trapèzes contenus dans les triangles [Pap15] (figure 1.15b). Mais ce *jittering* multiple ne prend cependant en compte qu'un nombre limité d'interdépendances spatiales entre les points distribués : la distribution de plusieurs impulsions à l'intérieur d'une cellule reste uniforme, sans considérer les positions des impulsions dans la cellule courante. Le *jittering* de grille ne prend également pas en compte les positions des impulsions situées sur les bords des cellules voisines, pouvant mener à des recouvrements d'impulsions (figure 1.15a). Une optimisation possible du *jittering* de grille est de remplacer les cellules rectangulaires par des polygones créés par tessellation [GDG12a] afin de contrôler plus précisément les dépendances spatiales (figure 1.16).

(a) (b)

FIGURE 1.16 – Distribution aléatoire dans des cellules tessélées [GDG12a]
(a) Une cellule de l'espace de distribution est tessellé de telle sorte qu'un pavage régulier de l'espace est toujours possible. (b) Des figures statistiques (ici, une fleur) sont distribués dans les différentes cellules

1.2.2 La subdivision de surface sur carte graphique

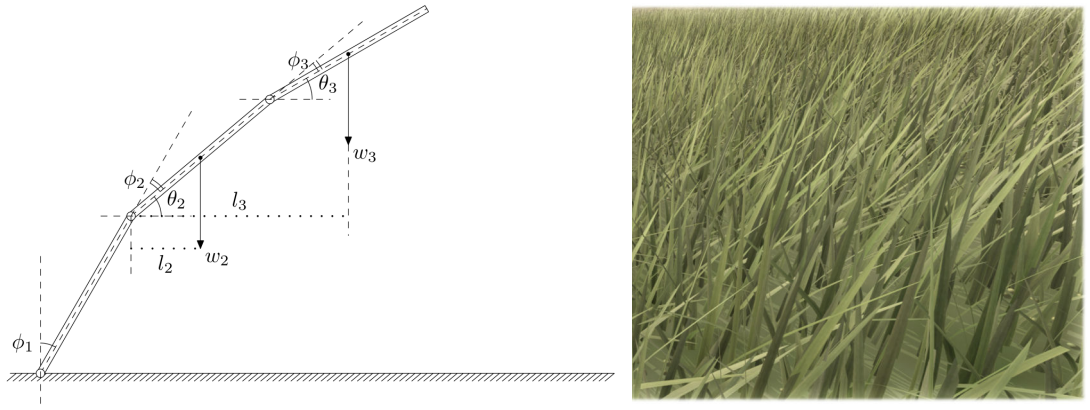
Une solution possible pour le rajout de détails géométriques est la subdivision de surface : un ensemble de primitives formant une surface est séparé en un ensemble de primitives plus grand. Les outils de subdivision sont particulièrement utiles pour obtenir un objet lisse à partir d'un maillage géométrique simplifié (aussi appelé *coarse mesh*). Un schéma de subdivision couramment cité dans la littérature est celui introduit par Catmull-Clark pour la génération de maillage à partir d'une surface de Bézier. Ce schéma considère une topologie arbitraire et subdivise le maillage suivant un ensemble de règles précises [Sta98, Nie13] en respectant cette topologie. Ce processus de subdivision peut être contrôlé pour générer automatiquement des déformations ou des objets de surface.

Pour réaliser une subdivision de maillage sur carte graphique, il est possible d'utiliser la *tessellation* de maillage : un maillage d'entrée est découpé en patchs (représenté par un ensemble de points de contrôle sur le maillage), pour lesquels l'opération de tessellation va générer un polygone triangulé dans lequel la densité de triangles est variable. Cette opération peut être réalisée en temps réel en utilisant les étapes de traitement programmables des cartes graphiques. Les cartes graphiques compatibles Direct3D11 et OpenGL4 possèdent des unités de traitement spécialisées dans la tessellation : à partir d'une géométrie décomposée en patchs de sommets, ces unités peuvent générer de nouveaux sommets et primitives par subdivision des arêtes du patch. Ces unités permettent une subdivision contrôlée par des programmes de tessellation (ou *tessellation shaders*) lors du rendu. Ces *shaders* interviennent sous deux formes dans le pipeline graphique :

- Les *Tessellation Control Shaders* définissent le traitement à effectuer en parallèle sur chaque sommet d'un patch (en indiquant par exemple les niveaux de subdivision).
- Les *Tessellation Evaluation Shaders* définissent le traitement à effectuer sur les nouveaux sommets générés par l'unité de tessellation.

Les *tessellation control shaders* en particulier préparent la subdivision des patchs selon deux niveaux : un niveau « interne » définit le nombre de subdivisions subies par l'intérieur du patch, tandis qu'un niveau « externe » définit le nombre de subdivisions subies par les arêtes externes du patch.

Ces programmes de tessellation permettent de contrôler finement la densité de géométrie présente sur la surface. Des détails de surface complexes peuvent ainsi être générés à la volée en utilisant par exemple un *displacement mapping* [NL13] : chaque nouveau sommet créé peut être déplacé selon un facteur hauteur le long de la normale recalculée pour recréer des déformations. En plus de minimiser la quantité de géométrie stockée, cette approche permet une génération adaptative de celle-ci. De nombreux critères tels que la courbure de la surface ou la profondeur de champ peuvent être pris en considération pour subdiviser le maillage sur des zones précises. Dupuy *et al.* [DHI*13] utilisent la distance de l'observateur à la surface des objets pour adapter le niveau de tessellation et recalculer une normale adéquate, suivant une distribution de micro-facettes particulières. Cette tessellation contrôlée leur permet d'équilibrer la quantité de géométrie affichée à la volée, sans artefacts de transition lors des changements de niveau de détails de la surface affichée.



(a) Bande de quadrilatères courbée

(b) Champ d'herbe généré par tessellation

FIGURE 1.17 – Génération d'herbe par tessellation de Papavasiliou [Pap15]
Les unités de tessellation sont utilisées pour créer une chaîne de quadrilatères courbée (a) lors du rendu.

Si la tessellation dynamique permet d'implémenter rapidement des techniques de rajout de déformations de surface, telles que le *displacement mapping*, elle permet également par extension de créer d'autres objets géométriques de manière procédurale lors du rendu. Papavasiliou [Pap15] utilise les programmes de tessellation pour créer la géométrie des brins d'herbe, à savoir une chaîne de quadrilatère, à la volée lorsque l'observateur se rapproche de la surface du terrain (figure 1.17). Pour chaque plante distribuée, un patch de tessellation ne contenant qu'un seul sommet (la racine de la plante) est créé. Chaque patch est ensuite subdivisé en segments de ligne en fonction de sa distance à l'observateur. Les différents segments sont associés à des masses pour calculer à la volée une courbure de la géométrie. Une dernière étape de tessellation transforme chaque segment en quadrilatère afin d'obtenir la géométrie finale de chaque brin.

1.2.3 Les textures procédurales

Le rajout de géométrie permet de détailler la forme d'un objet constitué d'un matériau, mais cette décoration peut également être réalisée par la définition de l'apparence de la surface grâce à une texture. De nombreuses méthodes permettent de créer ces textures à partir d'exemples et sont présentées plus en détail dans l'état de l'art proposé par Wei *et al.* [WLKT09]. Mais parmi celles-ci, les *textures procédurales* [EWM*98] permettent de générer l'apparence d'une texture à la volée pour un coût minime en mémoire. Les textures procédurales sont des fonctions renvoyant une couleur pour un point évalué et selon des paramètres passés en entrée de la fonction. Elles peuvent être créées sous de nombreuses formes, mais une construction courante est l'association d'une fonction de distribution procédurale de points et d'une fonction définissant un objet procédural. Ce type de texture permet d'obtenir en un temps d'évaluation constant une apparence de surface détaillée, quelle que soit la résolution de la surface affichée, tout en minimisant le besoin de stockage et le temps de modélisation des détails. L'amélioration du matériel graphique permet d'utiliser des textures procédurales de plus en plus complexes en temps réel, y compris des textures procédurales volumiques.

FIGURE 1.18 – Texture procédurale de marbre par Perlin [Per85].

Pour créer des motifs non répétitifs intégrant des variations aléatoires, ces textures procédurales peuvent s'appuyer sur des motifs de bruits précalculés ou sur des bruits procéduraux [LLC*10]. Ces derniers sont des fonctions produisant un motif non structuré aux propriétés fréquentielles particulières. Depuis l'apparition du bruit de Perlin [Per85] et de son motif de marbre (figure 1.18), les bruits procéduraux ont été très largement étudiés pour la reproduction de phénomènes naturels pouvant être approximés statistiquement. Ces bruits peuvent également être utilisés pour la génération procédurale de détails de surface tels que l'herbe : d'un point de vue visuel, un champ d'herbe vu de loin peut être perçu comme un motif non structuré pouvant être reproduit par une fonction de bruit (figure 1.19). Pangerl [Pan14] génère un motif de bruit utilisé pour représenter l'herbe par une méthode de *bavure* verticale évaluée en espace écran en post-traitement : la couleur d'un bloc de pixels du terrain est mélangée avec le bloc inférieur, un facteur de modulation selon l'axe horizontal permettant de ne réaliser cette opération que sur certaines lignes verticales de ces blocs de pixels, créant ainsi une forme de brins colorés dans l'image. Pour faciliter la création artistique de textures procédurales de surface, Gilet *et al.* [GDG12a] modifient la construction d'un bruit de convolution éparse en remplaçant le noyau de convolution par des figures statistiques composées de primitives paramétriques. Ces modèles sont distribués par *jittering*, subissent des déformations aléatoires, puis sont évalués. Tous ces calculs sont réalisés par une unique fonction de texture. Cette formulation permet de créer des motifs semi-aléatoires comportant des éléments de structure.

FIGURE 1.19 – Champ d'herbe procédurale calculée en espace écran [Pan14].

Les textures procédurales sont par nature indépendantes de leur dimension d'évaluation. Elles peuvent être rapidement étendues à la génération de textures solides [PCOS10], mais également aux textures volumiques. Une forme particulière de textures volumiques appelée *hypertextures* [PH89] permet de calculer une apparence très complexe par déformation d'un objet en modulant la densité de matière dans l'espace contenant celui-ci. Les fonctions de modulation utilisées par ces textures sont généralement basées sur des fonctions de bruit ou de turbulence. Gilet et Dischler [GD09] proposent une modification de la formulation des hypertextures de Perlin et Hoffert pour représenter des détails de surface. Ils introduisent pour cela une fonction de modulation de la surface de l'objet et une fonction de modulation de la couleur en fonction de cette déformation. Cette modification leur permet de générer des détails volumiques de surface, tels que les objets couverts d'herbe présentés dans la figure 1.20.

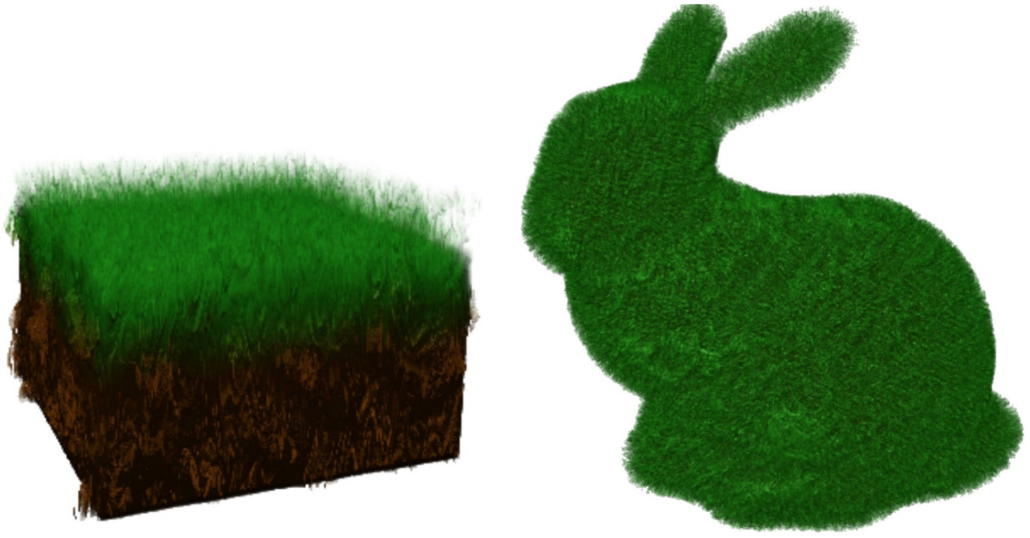


FIGURE 1.20 – Hypertextures d'herbe par Gilet et Dischler [GD09].

1.2.4 Synthèse

L'utilisation d'une distribution procédurale évaluable à la volée permet de créer une modélisation interactive des détails de surface. Mais pour créer des détails de surface non répétitifs, l'utilisation d'un nombre restreint d'objets discrets n'est pas suffisante. Dans le cas des motifs composés d'objets similaires ayant subit une faible quantité de variations pouvant être approximées statistiquement, le rajout d'éléments supplémentaires à distribuer et le rajout de transformations aléatoires permet de masquer la répétition de ces éléments. L'utilisation de représentations paramétriques pour créer procéduralement ces objets représente cependant une solution plus pérenne, permettant de générer ces variations pour chaque objet distribué.

La tessellation dynamique apporte une solution géométrique pour la génération de détails lors du rendu de la surface, permettant une génération à la volée complètement procédurale. Si cette géométrie de détails est associée à une texture semi-transparente, un ordonnancement des primitives rendu selon la distance à la caméra ou une stratégie de mélange adaptée peut cependant être nécessaire pour une évaluation correcte de la couleur. Dans le cas des brins, les tests d'opacité et de profondeur sur une unique passe de rendu ne permettent pas à eux seuls de réaliser correctement le mélange nécessaire à l'anti-aliassage des bords semi-transparents des textures de brins. Un autre problème est le contrôle de la génération des détails par cette approche : la création de géométrie par tessellation sur carte graphique nécessite des connaissances en programmation pour manipuler les *tessellation shaders*, en particulier selon l'interface de programmation utilisée pour accéder aux unités

Les textures procédurales permettent également de générer des détails de surface à la volée, sous la forme de textures surfaciques ou volumiques évaluables en chaque point de l'espace dans lequel elles sont évaluées. Parmi celles-ci, les textures basées sur les bruits procéduraux peuvent être utilisés pour reproduire statistiquement de nombreux phénomènes. Mais leur définition basée sur des propriétés fréquentielles rend leur paramétrage et le contrôle sur le motif difficiles pour un artiste : la connaissance de l'impact des modifications de l'espace fréquentiel sur le visuel d'un motif requiert un long apprentissage. Une solution proposée par certains bruits procéduraux est d'utiliser un processus d'analyse pour extraire ces paramètres d'un exemple, réduisant le travail de l'artiste à la réalisation d'un échantillon du motif voulu [LLC*10]. Mais un problème persiste : une fois les paramètres extraits, le contrôle directe sur le motif du bruit requière toujours une intervention dans l'espace fréquentiel. Certaines formes de textures procédurales se rapprochent de la formulation d'un bruit procédural tout en s'écartant de leur modélisation de l'espace fréquentiel [GDG12a], mais dans ce cas un paramétrage automatique devient également complexe. La solution étudiée dans cette thèse est un compromis entre ces deux approches : Nous introduisons un bruit procédural pouvant être paramétré automatiquement, mais permettant également à un artiste de modifier le motif généré non pas dans l'espace fréquentiel, mais directement dans l'espace visuel. Pour mieux appréhender cette contribution, un état de l'art plus approfondi des bruits procéduraux est présenté dans la section suivante.

1.3 Les bruits procéduraux pour la synthèse de textures

Les bruits sont un outil fondamental pour la production de nombreux phénomènes : la fumée, les nuages, les vagues et bien d'autres phénomènes naturels peuvent être approximés par des motifs aléatoires et non structurés. Cet outil se retrouve sous différentes formes et formulations dans de nombreux logiciels de modélisation, par exemple sous la forme de textures ou encore de fonctions. Parmi ces dernières, les bruits procéduraux sont intensivement étudiés pour la production de textures procédurales non-structurées toujours plus complexes. Ces bruits peuvent aider les artistes à la modélisation interactive de textures irrégulières quasi infinies, mais les bruits procéduraux les plus récents permettent également d'intégrer certaines formes de structures dans un motif irrégulier.

Dans les domaines liés à l'image, les bruits sont couramment définis comme des signaux caractérisés par leur spectre de puissance dans le domaine fréquentiel. Pour reproduire le visuel correspondant à ce spectre en un point, une fonction de bruit peut être construite comme une distribution de valeurs autour de ce point, les valeurs ou leurs positions pouvant être créées par un générateur de nombres aléatoires. Les premiers bruits procéduraux proposés dans la littérature utilisent une interpolation de valeurs ou de gradients aléatoires situés sur une grille régulière autour du point évalué. Une autre approche possible est de réaliser la convolution d'une distribution avec un noyau d'évaluation. Cette convolution est plus généralement approximée par la somme des contributions des noyaux distribués autour du point évalué. Ces deux familles de bruits procéduraux, respectivement appelées bruits d'interpolation et bruits de convolution, sont particulièrement étudiées pour leurs performances d'évaluation par pixel compatibles avec les applications temps réel. Ces bruits permettent un contrôle du spectre sous différentes formes, par le biais de paramètres régissant son aspect fréquentiel. Certains d'entre eux proposent également une méthode d'analyse pour l'extraction automatique de ces paramètres à partir d'un spectre donné en entrée.

Nous réintroduisons dans la partie suivante les différentes fonctions de bruits pour la synthèse de textures procédurales. Nous récapitulons en premier lieu la définition et les propriétés fréquentielles attribuées aux bruits procéduraux. Nous rappelons ensuite les différents bruits procéduraux existants sous forme de fonction selon leur type d'évaluation, à savoir les bruits d'interpolation et les bruits de convolution. Nous n'abordons pas dans cette partie le filtrage des bruits, pour nous concentrer principalement sur les possibles contrôles de l'apparence d'un motif produit par un bruit procédural. Pour un état de l'art plus complet sur l'ensemble des bruits procéduraux, nous invitons le lecteur à lire l'article de Lagae *et al.* [LLC*10].

1.3.1 Les bruits et l'espace fréquentiel

Les bruits sont couramment cités comme les générateurs de nombres aléatoires de l'informatique graphique. Ces signaux ne présentent aucune structure visuelle, mais ils possèdent une apparence fréquentielle particulière, décrite en termes de magnitude et de phase pour un ensemble de sinusoïdes permettant de reproduire ce signal à l'identique. Plus spécifiquement, ces motifs et ces bruits sont généralement caractérisés par leur **PSD**, ou *densité spectrale de puissance* ,aussi appelée spectre de puissance.

Une des premières définitions des bruits pour l'informatique graphique est donnée par Perlin [Per85], considérant un bruit comme une approximation d'un bruit blanc limité sur les fréquences d'une octave, et étant statistiquement invariant par rotation et translation. Cette définition introduit l'idée de la production d'un motif aléatoire aux propriétés fréquentielles et statistiques précises. Une définition est plus récemment donnée par Lagae *et al.* [LLC*10], pour qui un bruit est un processus aléatoire normal et stationnaire, dont le spectre de puissance peut être contrôlé soit par édition du spectre, soit par la somme d'instances indépendantes de bruits passe-bande. Le spectre de puissance d'un bruit suivant la définition précédente correspond à la transformée de Fourier de sa fonction d'auto-corrélation. Cette dernière définie la valeur attendue par le produit de ce bruit évalué en deux positions distinctes.

Pour extraire ce même spectre d'une image, la **DFT** ou *Transformée de Fourier Discrète* est l'outil de référence utilisé dans de nombreux domaines liés à l'image ou aux traitements de signaux, en particulier pour obtenir les spectres de magnitude et de phase d'un signal discret. Le spectre de puissance d'une image est défini comme le carré du spectre de magnitude. Cette transformée traduit un signal discret s représenté par un ensemble de N valeurs en un ensemble S de N coefficients de Fourier. Cette opération est inversible, les valeurs spatiales de l'image pouvant être recalculées à partir de ses informations fréquentielles. La DFT et son inverse sont plus généralement formulées sous une forme complexe par l'équation suivante :

$$S[k] = \sum_{n=0}^{N-1} s[n] e^{-2i\pi k \frac{n}{N}} \quad ; \quad s[n] = \frac{1}{N} \sum_{k=0}^{N-1} S[k] e^{-2i\pi n \frac{k}{N}} \quad ; \quad k \in [0, N[\tag{1.4}$$

Chaque coefficient complexe $S[k]$ représente une fréquence correspondant à une sinusoïde caractérisée par une magnitude $M[k]$ et une phase $\phi[k]$. Cette magnitude et cette phase correspondent au module et à l'argument de $S[k]$:

$$M[k] = \sqrt{Re(S[k])^2 + Im(S[k])^2} \qquad \phi[k] = \arctan\left(\frac{Im(S[k])}{Re(S[k])}\right)$$
$$Re(S[k]) = M[k]\cos(\phi[k]) \qquad Im(S[k]) = M[k]\sin(\phi[k])$$

L'ensemble des phases et celui des magnitudes forment respectivement les spectres de magnitude et de phase. Pour l'analyse spectrale et le traitement de l'images en temps réel, une version particulière de la **DFT** appelé *Transformée de Fourier rapide* (ou **FFT** pour *Fast Fourier Transformation*) est généralement utilisée pour la transformation en espace

fréquentiel. Si l'algorithme de Cooley-Tuckey [BNW*95] est souvent la méthode de référence pour le calcul de la **FFT**, des implantations de la DFT optimisées pour carte graphique permettent également une évaluation rapide du spectre et sont énumérées par Govindaraju *et al.* dans [GLD*08].

Pour produire un motif de bruit à partir d'un spectre de puissance, une possible approche est de réaliser une **DFT** inverse de ce spectre en générant des phases aléatoires pour chaque fréquence. Mais le temps de calcul d'une **DFT** inverse, ou même d'une **FFT** inverse, devient rapidement prohibitif avec l'augmentation du nombre de fréquence pris en compte. Le fait que cette approche considère également les fréquences ne contribuant pas au motif généré rend cette approche d'autant moins attrayante pour la production en temps réel de textures de grande envergure ou dont la résolution est élevée. *A contrario*, si ses paramètres lui permettent de reproduire ce spectre, un bruit procédural peut reproduire un motif continu à l'infinie et ce en un temps significativement plus court. Suivant la définition précédente de Lagae *et al.*, un bruit *procédural* présuppose les caractéristiques suivantes en plus des caractéristiques d'un bruit :

- Un bruit procédural est un programme extrêmement compact, stocké sous la forme d'un programme de quelques kilo-octets.
- Un bruit procédural est continu, multi-résolution, et ne dépend pas d'un ensemble discret de données pour son évaluation.
- Un bruit procédural est non périodique, et peut s'étendre infiniment dans son espace d'évaluation.
- Un bruit procédural est paramétrique, ses paramètres lui permettant de générer un grand nombre de motifs différents.
- Un bruit procédural est accessible en n'importe quelle position, et son évaluation est constante quelle que soit cette position et indépendante d'une évaluation précédente.

Les bruits procéduraux sont généralement classés selon trois catégories liées à leur construction : les bruits de grilles de gradients, les bruits explicites et les bruits de convolution éparse. Parmi ces trois familles, les bruits de grilles de gradients et les bruits de convolution peuvent être créés sous formes de fonctions évaluables par point sans précalcul d'un motif de bruit : les bruits de grilles de gradients reposent sur la distribution de gradients sur une grille régulière, ces gradients étant générés par un **PRNG**, et les bruits de convolution reposent sur la distribution aléatoire ou régulière de noyaux d'évaluation renvoyant une valeur en fonction d'une distance ou d'un vecteur. Les bruits explicites se basent sur certaines données de bruit précalculées, telles qu'une texture de bruit prégénérée en utilisant une **DFT** inverse, et dépendent donc d'un ensemble discret de données.

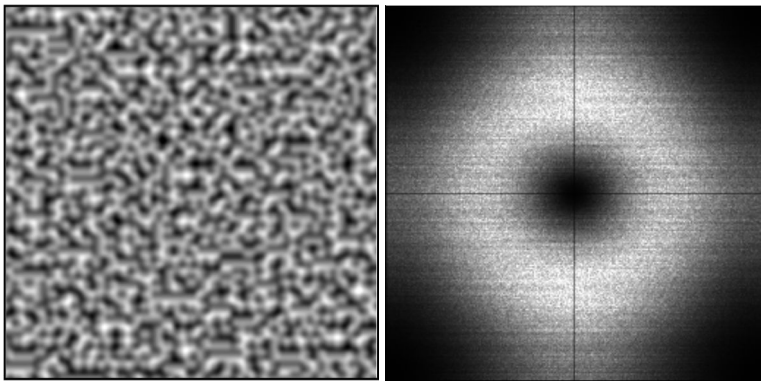
Nous choisissons de distinguer les différentes fonctions de bruits procéduraux évaluables par point en deux grandes familles, présentées ci-après, selon leur type d'évaluation : les bruits d'interpolation et les bruits de convolution.

1.3.2 Les fonctions de bruits d'interpolation

Ces bruits se basent sur l'interpolation de données aléatoires distribuées sur une grille régulière : chaque sommet de la grille est associé à une donnée générée par un processus pseudo-aléatoire, l'évaluation en une position quelconque se faisant alors par interpolation entre les données situées aux sommets les plus proches du point sur la grille. Le bruit de Perlin [Per85, Per02], souvent cité comme le précurseur des textures procédurales, est le premier des bruits apparus de cette catégorie, se basant sur une interpolation entre huit gradients situés sur les sommets les plus proches du point d'évaluation dans une grille rectiligne. Ces gradients sont choisis aléatoirement parmi une liste prédéfinie : chaque sommet de la grille est utilisé dans une fonction de hachage pour décorréler les coordonnées de la grille de l'indice des gradients aléatoires stockés dans un tableau. La liste des gradients originellement proposée par Perlin est un tableau de douze vecteurs unitaires partant du centre d'un cube vers le centre de ses arêtes. Dans son implantation, une interpolation cubique est réalisée par une interpolation linéaire dont le facteur d'interpolation par axe est modifié selon une courbe de décroissance :

$$f(x) = 6x^5 - 15x^4 + 10x^3 \tag{1.5}$$

Le bruit de Perlin, supposé être un bruit passe-bande isotrope, génère cependant un spectre de puissance contenant de nombreux dépassements de la bande de fréquences ciblée, et de nombreux artefacts d'anisotropie spectrale (figure 1.21).

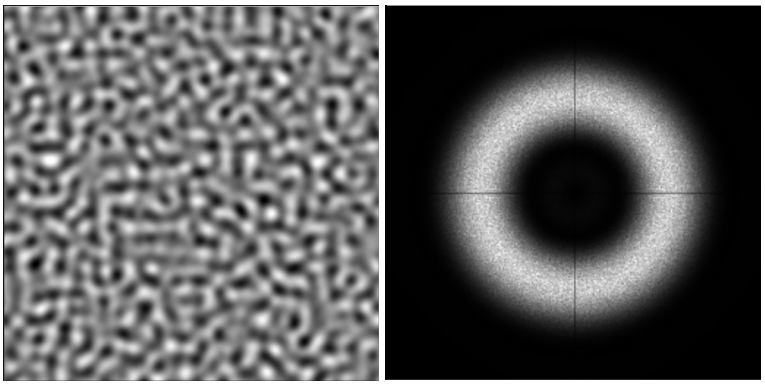


(a) Bruit (b) PSD ([LLC*10])

FIGURE 1.21 – Bruit de Perlin [Per85]

Des corrections ont été proposées pour créer des bruits de grilles de gradients de meilleure qualité dans différentes implémentations. Ces améliorations portent sur les trois composants de ce bruit : la fonction de hachage, la construction du tableau de gradient et la fonction d'interpolation. Olano [Ola05] propose dans son implémentation CMOS de construire la grille de gradients à partir des coins d'un cube unitaire. Pour corriger les différents problèmes du bruit de Perlin originel, le bruit de gradient amélioré de Spjut *et al.* [SKB09] introduit une nouvelle fonction de hachage et un tableau de gradients plus important ainsi qu'une fonction d'interpolation associée à un filtre radial. Ces corrections permettent de réduire les artefacts d'anisotropie et améliorent le filtrage des

fréquences (figure 1.22).



(a) Bruit (b) PSD ([LLC*10])

FIGURE 1.22 – Le bruit de gradient amélioré de Spjut *et al.* [SKB09]

Parmi les bruits d'interpolation existants, certains ont la particularité de se focaliser sur la création de textures de bruits animées reproduisant des systèmes de simulation physique. Le défi posé par ces textures est de conserver certaines propriétés telles que le spectre du motif et certaines caractéristiques visuelles, tout en respectant les propriétés physiques du phénomène représenté et en conservant une cohérence temporelle. Pour représenter des fluides, Perlin et Neyret [PN01] présentent le bruit de flux, une évolution du bruit de Perlin pour l'animation de fluides reposant sur une déformation de la texture en fonction d'un champ de vecteurs, autrement appelée l'*advection* de texture. Un problème inhérent à ce bruit est cependant qu'il ne respecte pas strictement les propriétés des fluides et que des distorsions importantes peuvent apparaître. Une correction possible est proposée par Yu *et al.* [YNBH11], utilisant une formulation lagrangienne du champ de vecteur pour préserver le spectre de puissance de la texture et le champ de vélocité au cours du temps (figure 1.23). Le bruit de courbe [BHN07] permet également de créer des fluides incompressibles, utilisant un bruit de Perlin associé à des champs de potentiel pour créer un champ de vélocité turbulent.

1.3.3 Les fonctions de bruits de convolution

Un bruit de convolution est construit par la convolution d'un noyau d'évaluation avec une distribution d'impulsions, chaque impulsion associant une position à une intensité. Dans la pratique, ces bruits reposent sur la distribution et la pondération aléatoires de noyaux de convolution autour du point évalué, l'évaluation du bruit se faisant par la somme des contributions en ce point des noyaux distribués. Le bruit de convolution éparse introduit par Lewis [Lew89] est l'initiateur de cette famille de bruits (figure 1.24).

Les bruits de convolution présentent un avantage certain quant à la modélisation d'un spectre précis : la théorie de la convolution énonce que le spectre de puissance d'un signal issu de la convolution de plusieurs signaux est le produit des spectres de puissance de

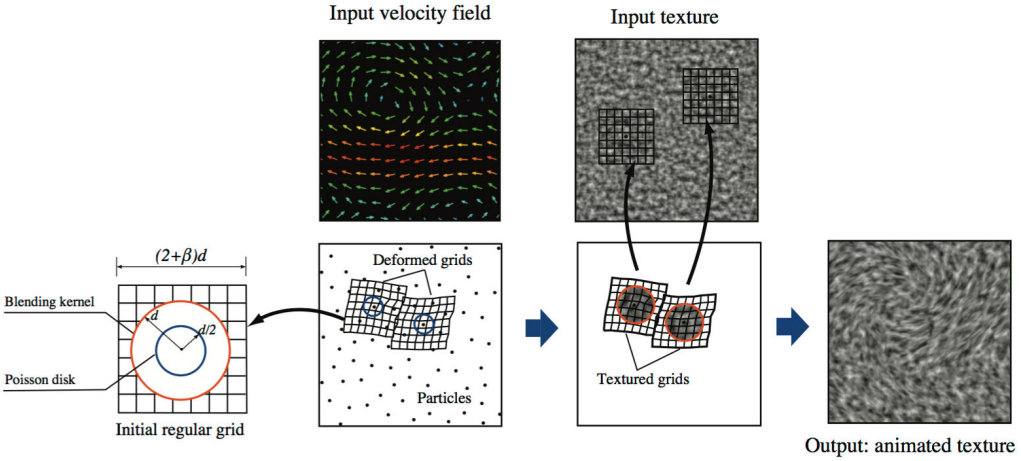


FIGURE 1.23 – Advection lagrangienne d'une texture de bruit de Perlin par Yu *et al.* [YNBH11].

ces signaux. Le spectre de puissance d'un bruit de convolution est alors égal à la multiplication du spectre du noyau par le spectre de sa distribution. Pour faciliter la modélisation spectrale, une solution est d'utiliser une distribution dont le spectre est constant pour se focaliser sur l'édition du spectre du noyau. La formulation de ce type de bruit évalué en un point en deux dimensions est généralement la suivante :

$$N(x,y) = \int\int \gamma(u,v)k(x-u,y-v)dudv \tag{1.6}$$

Où $k(x,y)$ est le noyau de convolution. Le bruit de convolution éparse (et de nombreux autres bruits de convolution) utilise une distribution de Poisson ($\gamma(u,v)$) pour générer des impulsions δ à des positions aléatoires (x_k,y_k) et à des intensités aléatoires a_k :

$$\gamma(x,y) = \sum_k a_k \delta(x-x_k, y-y_k) \tag{1.7}$$

Une distribution de Poisson implique un écart minimal entre les impulsions, comme précisé dans la section 1.2.1, mais également un spectre de puissance constant. Pour évaluer cette fonction de bruit en un point quelconque, il n'est pas nécessaire d'évaluer les noyaux associés à l'ensemble des impulsions : les noyaux étant supposés avoir un support fini, c'est-à-dire être tronqués à partir d'une distance déterminée, seuls les noyaux suffisament proches du point évalué sont nécessaires. L'évaluation peut être accélérée par l'utilisation d'une grille virtuelle dont la taille des cellules est égale à la taille des noyaux, pour ne prendre en compte uniquement que les impulsions situées dans la cellule du point évalué et dans les cellules voisines. Le calcul de cette convolution est souvent approximé par la somme des contributions du noyau de convolution centré en chaque position des impulsions i couvrant le point évalué, avec une amplitude mise à l'échelle :

$$N(x,y) \approx \sum_i w_i k(x-x_i, y-y_i) \tag{1.8}$$

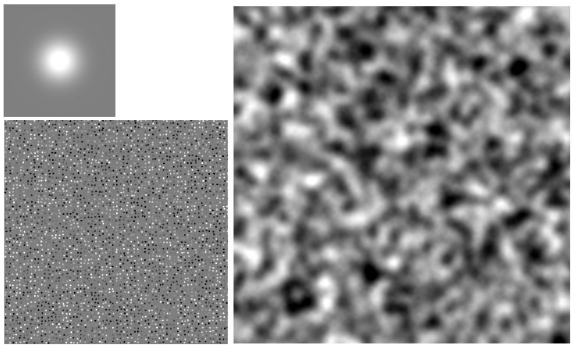


FIGURE 1.24 – Bruit de convolution éparse de Lewis [Lew89].
Le motif de droite est produit par la convolution d'une gaussienne tronquée (en haut à gauche) avec une distribution d'impulsions aléatoirement positionnées et pondérées (en bas à gauche).

Proche du bruit de convolution de Lewis, le bruit de spot [vW91] est initialement conçu pour la création de textures stochastiques pour la visualisation de champs de vecteurs. Ce bruit repose sur une simplification de la formulation du bruit précédent en considérant qu'une fonction de bruit peut être créée par la distribution aléatoire d'impulsions associées à un *spot*, très similaire au noyau d'évaluation du bruit de convolution éparse :

$$f(t) = \sum_i a_i h(t - t_i) \tag{1.9}$$

Van Wijk avance que le spectre de puissance du signal produit par cette formulation correspond au spectre du *spot*. Il constate également qu'une partie de la structure du *spot* utilisé est reportée sur le motif de bruit produit (figure 1.25).

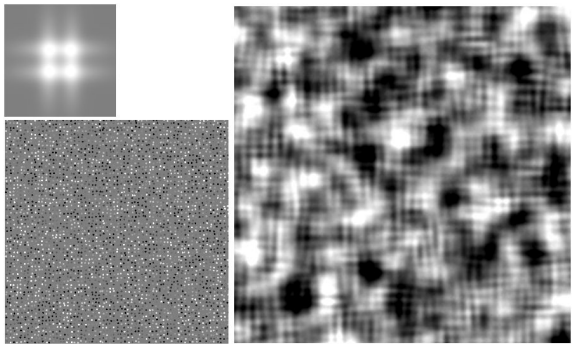


FIGURE 1.25 – Bruit de spot de Van Wijk [vW91].
La forme particulière du *spot* en haut à gauche ajoute une structure similaire au motif de bruit produit.

Le principal reproche fait au bruit de convolution éparse et au bruit de spot est également l'un de leurs atouts : le choix du noyau est laissé à l'appréciation de l'utilisateur. Idéalement, le noyau utilisé dans ces deux bruits doit avoir un support spatial fini, c'est-à-dire une intensité maximale au centre du noyau puis diminuant jusqu'à être nulle à partir d'une distance d par rapport au centre. Lewis [Lew89] recommande pour cela un noyau de convolution stocké sous la forme d'un élément de texture construit comme la multiplication d'un échantillon de motif par une enveloppe gaussienne isotrope tronquée. Un autre problème est le paramétrage de la densité d'impulsion par zone de distribution, qui doit être suffisamment élevée pour éviter l'apparition de creux.

D'autres fonctions de bruits reprennent le principe du bruit de convolution mais en précisant cette fois un noyau particulier ou bien une méthodologie pour la construction de noyaux. Lagae *et al.* [LLDD09] introduisent le noyau de Gabor pour la création de motifs aux spectres de puissance anisotropes et passe-bande. Ce noyau est conçu comme la multiplication dans le domaine spatial d'une enveloppe gaussienne et d'une harmonique sphérique :

$$g(x,y) = K\mathrm{e}^{-\pi a^2(x^2+y^2)}\cos[2\pi F_0(x\cos\omega_0 + y\sin\omega_0)] \tag{1.10}$$

Le spectre généré par ce noyau est anisotrope, mais un spectre isotrope peut être également obtenu en générant pour chaque impulsion une valeur aléatoire pour ω_0 comprise dans l'intervalle $[0, 2\pi[$ (figure 1.26). L'avantage majeur de ce noyau est lié à son expression analytique $G(f_x, f_y)$ dans le domaine fréquentiel, permettant une visualisation et une modification directes du spectre pour l'édition :

$$\begin{aligned} G(f_x,f_y) = \frac{K}{2a^2}\{&\mathrm{e}^{\frac{-\pi}{2a}[(f_x - F_0\cos\omega_0)^2 + (f_y - F_0\sin\omega_0)^2]} \\ &+ \mathrm{e}^{\frac{-\pi}{2a}[(f_x + F_0\cos\omega_0)^2 + (f_y + F_0\sin\omega_0)^2]}\} \end{aligned} \tag{1.11}$$

Une amélioration de ce noyau pour la création et le filtrage de bruits solides (c'est-à-dire reproduisant un matériau de sculpture sur un objet) est également donnée par Lagae *et al.* [LD11], prenant en compte une phase pour évaluer correctement les coupes du matériau sans discontinuités apparentes :

$$g(\mathbf{x}; a, \omega, \phi) = \mathrm{e}^{-\pi a^2|\mathbf{x}|^2}\cos(2\pi\mathbf{x}\cdot\omega + \phi) \tag{1.12}$$

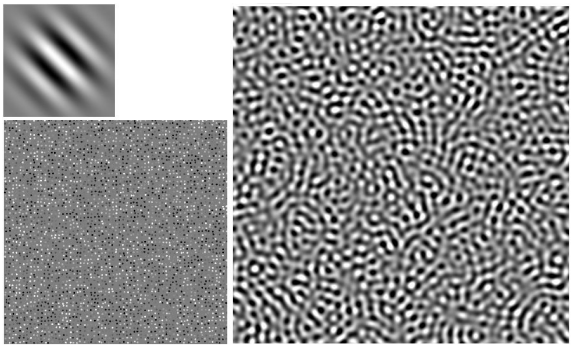


FIGURE 1.26 – Bruit de Gabor de Lagae *et al.* [LLDD09].
Pour chaque impulsion, une orientation aléatoire de l'harmonique permet de générer un motif correspondant à un spectre passe-bande isotrope.

Pour créer des motifs plus complexes, Gilet *et al.* [GDG12b] proposent d'utiliser un noyau basé sur un sinus cardinal limité (c'est-à-dire égal à 0 si $x > d$) comme primitive spectrale pour la création de bruit multi-noyaux :

$$\lfloor sinc(x) \rfloor_d = \frac{sin(\pi x)}{\pi x} \tag{1.13}$$

Ce noyau produit un spectre de puissance rectangulaire, pouvant être utilisé pour décomposer un spectre de puissance en différentes bandes de fréquences.

Parmi les bruits de convolution, des exceptions n'utilisant pas une distribution aléatoire d'impulsions existent. Le bruit de simplex [OHH*02] est une évolution de bruit de Perlin, pour laquelle l'interpolation de gradients distribués sur une grille cubique est remplacée par une somme de gradients associés à une fonction d'atténuation et distribués sur une grille de simplexes. Par rapport à la grille cubique, la grille de simplexes permet de réduire la complexité de calcul, le nombre de gradients à évaluer passant pour une dimension N de (2^N) à $(N+1)$. Le remplacement de l'interpolation par une somme de contributions multipliées par une fonction d'atténuation réduit également le nombre d'instructions pour l'évaluation. Le bruit local à phase aléatoire de Gilet *et al.* [GSV*14] n'utilise également plus une distribution aléatoire d'impulsions, mais un ensemble de bruits locaux distribués sur une grille rectiligne (figure 1.27) :

$$n(x) = \sum_{i=1}^{I} w\left(\frac{||x - x_i||}{\Delta}\right) \sum_{j=1} A_{i,j} cos(2\pi f_{i,j} \cdot x + \rho_{i,j}) \tag{1.14}$$

Chacun des bruits distribués est construit comme une somme de sinusoïdes contrôlées par une magnitude A, une fréquence f et une phase ρ. Ce modèle de bruit permet de reproduire des motifs gaussiens et non gaussiens, ces derniers présentant des éléments structurels dont les caractéristiques se situent à la fois dans le spectre de puissance et dans le spectre de phase du motif. Les fréquences de ces structures sont identifiées par Gilet *et al.* comme étant les zones du spectre de puissance contenant la plus grande densité d'énergie par fréquence. Un motif non gaussien peut alors être reproduit par ce bruit en réutilisant les phases de ces fréquences.

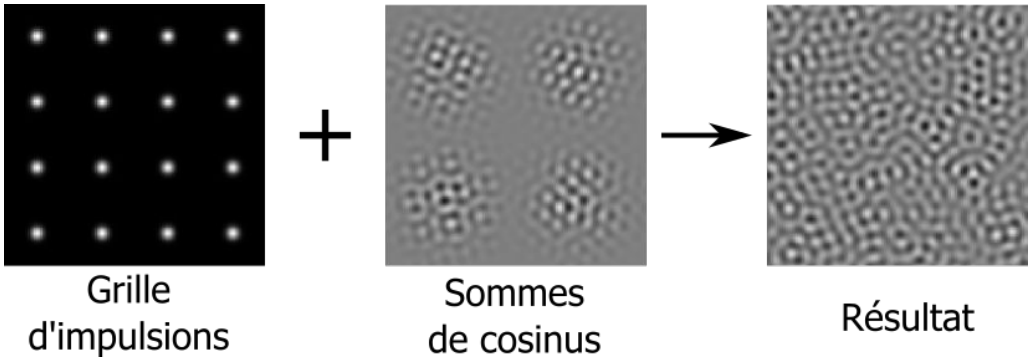


FIGURE 1.27 – Bruit local à phase aléatoire de Gilet *et al.* [GSV*14]

1.3.4 Modeler le motif produit par un bruit procédural

Les bruits procéduraux sont plus couramment considérés comme un outil de modélisation intégré dans le processus de création d'une texture procédurale. Ces bruits sont souvent référencés comme des fonctions de base de texture, c'est-à-dire des fonctions définissant la structure visuelle sous-jacente d'une texture procédurale : une fonction de bruit renvoie une intensité entre 0 et 1 pour chaque point évalué, cette intensité étant ensuite associée à une couleur renvoyée par une carte ou une fonction. L'apparence de la texture créée en utilisant une fonction de bruit reste fortement dépendante du motif généré par le bruit lui-même, ce motif pouvant fortement varier selon son paramétrage.

Pour la modélisation de ce motif, certaines fonctions de bruit définissent des paramètres agissant directement sur le spectre de puissance, là où d'autres permettent de modifier certaines informations spatiales. Des bruits procéduraux plus récents se focalisent sur l'extraction de paramètres à partir d'un exemple pour limiter la manipulation du spectre de puissance par l'utilisateur.

Suivant les définitions précédentes de Perlin et Lagae *et al.*, la possibilité pour un bruit procédural de produire un motif particulier dépend principalement de sa capacité à reproduire son spectre de puissance. La reproduction d'un motif aléatoire élémentaire, c'est-à-dire dont le spectre de puissance est contenu dans une bande de fréquences précise, nécessite un bruit procédural pouvant produire un spectre de puissance limité à cette même bande de fréquences. À l'aide d'un tel bruit, il est possible de recomposer des spectres de puissance complexes par l'addition pondérée de bruits passe-bande couvrant les différentes bandes de fréquences ciblées.

Les bruits se basant sur la convolution permettent de contrôler le spectre de puissance en modifiant celui du noyau s'ils utilisent une distribution particulière : comme énoncé dans la partie précédente, la convolution d'une distribution de Poisson et d'un noyau d'évaluation produit un spectre de puissance équivalant à celui du noyau d'évaluation modifié par un facteur d'échelle. Pour construire un bruit procédural permettant l'édition interactive du spectre, il est possible d'utiliser des noyaux de convolution dont le spectre est connu et paramétrable. Le noyau de Gabor permet par exemple à Lagae *et al.* [LLDD09] de fournir une interface visuelle pour aider les artistes à l'édition d'un spectre de puissance : un ensemble de noyaux peut être paramétré interactivement pour couvrir une bande de fréquences, chaque noyau définissant un nouveau bruit de Gabor reproduisant cette bande de fréquences. Le noyau utilisé par Gilet *et al.* [GSV*14] dans l'équation 1.14 pourrait également être contrôlé spectralement par un artiste pour la création de motifs aléatoires, en dessinant par exemple directement le spectre de puissance pour définir les fréquences et les amplitudes du noyau.

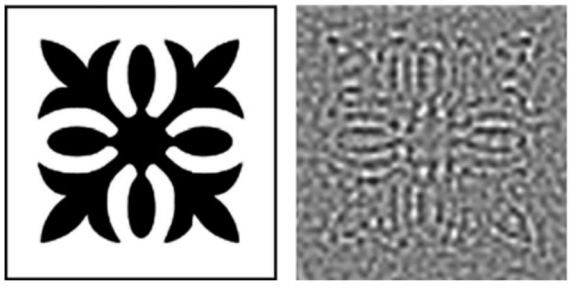
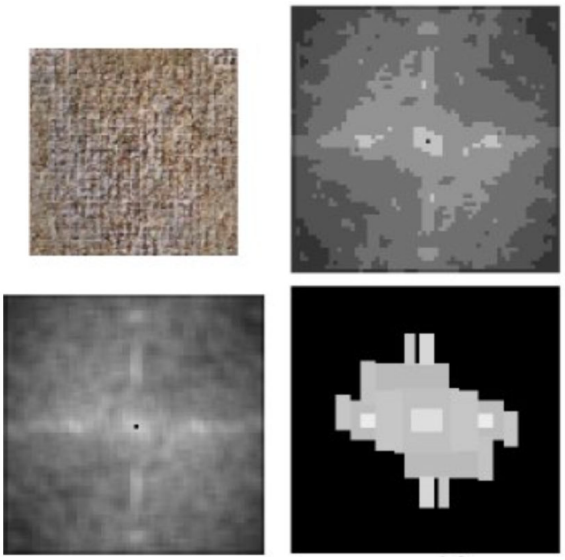


FIGURE 1.28 – Contrôle spatial par Yoon et Lee [YL08]
La contrainte représentée par le masque de gauche crée un motif de bruit dans lequel la structure du masque est transposée.

Le paramétrage spectral n'est cependant pas une approche facilitant la création artistique d'un motif : la corrélation entre les espaces fréquentiel et visuel n'est pas évidente pour un artiste. Certains bruits favorisent pour cela une forme d'édition spatiale des valeurs de leur signal. Les bruits tels que le bruit de flux [PN01] et le bruit de courbe

[BHN07] sont un possible exemple d'édition spatiale, ces bruits utilisant des champs de vecteurs ou de potentiel pouvant être définis par un artiste pour guider le motif. Ces bruits sont cependant limités à la création de textures très spécifiques, en l'occurrence des textures de fluides, et un bruit de Perlin est toujours utilisé comme fonction de base de texture. D'autres méthodes proposent de modifier directement certaines valeurs du bruit pour influencer le générateur aléatoire sous-jacent. Yoon et Lee [YL08] introduisent dans cette optique un bruit contraint par un masque, cette contrainte ayant pour effet de transposer la structure du masque dans le motif de bruit (figure 1.28).



(a) Motif et PSD (b) Décomposition rectangulaire

FIGURE 1.29 – Décomposition de la PSD pour le paramétrage automatique du bruit à noyaux multiples [GDG12b]
La **PSD** d'un motif (a) est découpé en région rectangulaire (b), le spectre de ces régions pouvant être approximé par des noyaux de Gabor ou des noyaux *sinc*.

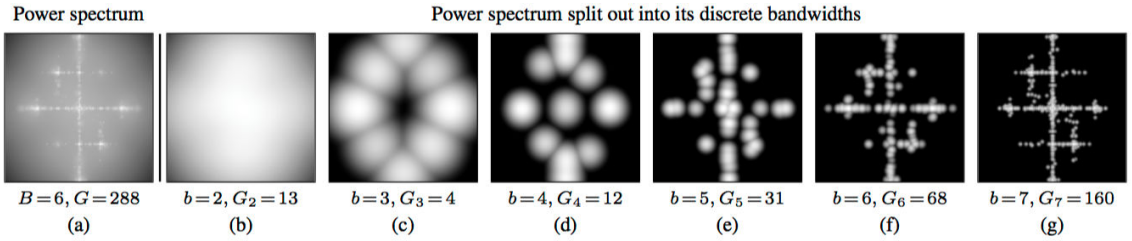


FIGURE 1.30 – Décomposition de la PSD pour le paramétrage automatique du bruit de Gabor [GLLD12]
Une PSD (a) est décomposée en différentes bandes de fréquences couvertes par des gaussiennes spectrales (b-g) correspondant aux paramètres de plusieurs bruits de Gabor.

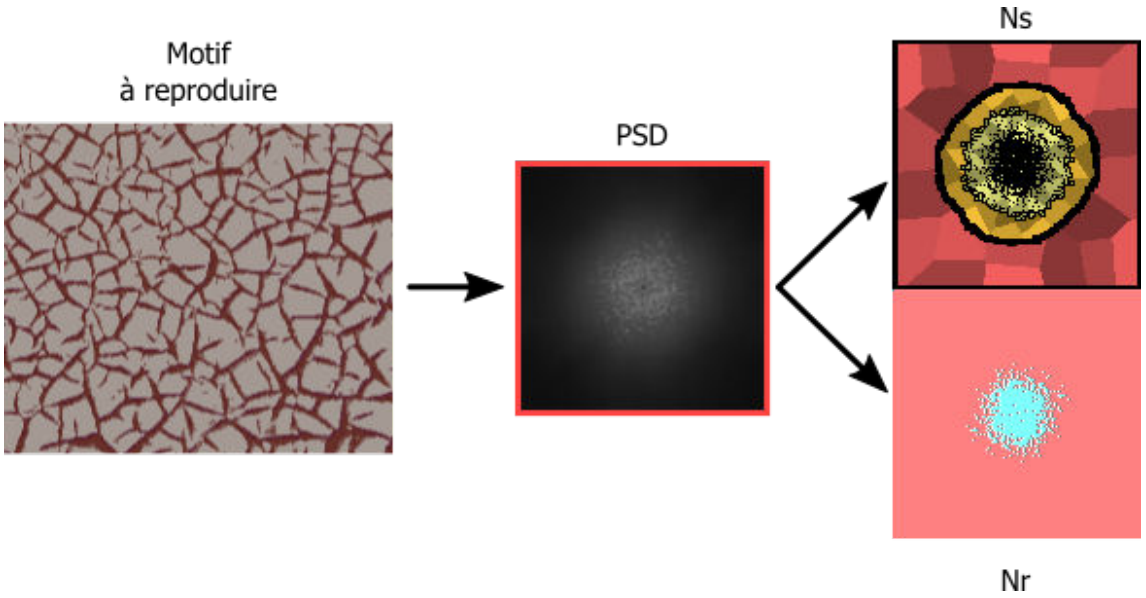


FIGURE 1.31 – Décomposition de la PSD pour le paramétrage automatique du bruit local à phase aléatoire [GSV*14]
La **PSD** d'un motif semi-structuré (au centre) est décomposée en un ensemble de bruits Ns utilisant des phases aléatoires, plus un bruit local Nr correspondant aux fréquences de la strate décrivant les structures du motif et pour laquelle les phases sont conservées.

Des bruits procéduraux plus récents fournissent des processus d'analyse et d'extraction de paramètres depuis un exemple pour simplifier l'édition de motifs et de textures. Gilet *et al.* [GDG12b] proposent de décomposer le spectre de puissance d'un exemple en un sous-ensemble de spectres de puissance rectangulaires selon des seuils de magnitude (figure 1.29). Ces différents spectres sont ensuite approximés à l'aide de noyaux de Gabor (éq. 1.10) ou de noyaux basés sur un sinus cardinal (éq. 1.13). Galerne *et al.* [GLLD12] proposent dans cette même optique une extension du bruit de Gabor dans laquelle les paramètres sont extraits par une décomposition du spectre en gaussiennes spectrales (figure 1.30). Le spectre est discrétisé en plusieurs bandes de fréquences, chacune de ces bandes étant ensuite décomposée en gaussiennes spectrales, une gaussienne correspondant aux paramètres d'un noyau de Gabor augmenté d'une phase (éq. 1.12). Une autre approche est proposée par Gilet *et al.* [GSV*14] pour décomposer un spectre de puissance en sous-parties pouvant être approximées par des bruits (figure 1.31). Le spectre est découpé dans un premier temps en différentes strates possédant toutes une quantité d'énergie égale. Pour limiter les fuites d'énergie entre les différentes strates du spectre de puissance, ces dernières sont réduites par érosion morphologique. Chaque strate est ensuite redécoupée en sous-strates, dans lesquelles une fréquence sera aléatoirement tirée lors de la reconstruction de la texture par le bruit. Contrairement aux méthodes précédentes, celle de Gilet *et al.* considère également une strate centrale composée des basses fréquences contenant un certain pourcentage de l'énergie totale du spectre de puissance. Pour cette strate supplémentaire, les phases sont conservées pour pouvoir reproduire des éléments de structure.

1.3.5 Synthèse

Les bruits procéduraux tendent à favoriser l'analyse du spectre de puissance pour la reproduction de motifs. Un spectre de puissance complexe pouvant être reproduit par une somme de bruits passe-bande couvrant les différentes fréquences, l'intérêt d'un bruit procédurale se reporte sur trois aspects : la fiabilité de sa limite en bande de fréquence, ses possibilité de contrôle de son spectre élémentaire et sa vitesse d'évaluation. Si les bruits d'interpolation sont souvent favorisés pour leur performance d'évaluation, les bruits de convolution peuvent s'avérer plus intéressants autant pour la qualité du spectre de puissance produit qu'en termes de contrôle, et ce pour une perte de performance acceptable pour le rendu temps réel.

La définition des bruits axée sur une caractérisation par le spectre de puissance a conduit la communauté scientifique vers la création de bruits procéduraux au spectre de puissance paramétrable. Mais le paramétrage spectral complexifie l'édition artistique interactive autant pour la création de motifs 2D que celle de motifs 3D : bien que les aspects fréquentiels et visuels soient interdépendants, il est difficile pour un artiste de prévoir quel impact une modification du spectre de puissance aura sur l'apparence visuelle du motif. Dans le cas des motifs procéduraux 3D, aucun des bruits cités précédemment ne proposent une interface permettant une telle édition.

Une précision est également apportée par Lagae *et al.* [LLC*10] et rappelée par Gilet *et al.* [GSV*14] : la caractérisation d'un bruit par le seul spectre de puissance signifie que seuls des motifs gaussiens (sans aucun élément de structure) peuvent être créés par une telle fonction de bruit. Les motifs aléatoires non gaussiens, présentant certaines régularités structurelles ou certains éléments de structure, nécessitent une caractérisation plus approfondie ou une approche d'édition complémentaire à celle du spectre de puissance. Gilet *et al.* considèrent que les éléments de structure présents dans un motif semi-aléatoire sont dépendants des phases correspondant à leurs fréquences, et réussissent à reproduire ces éléments particuliers en utilisant ces phases dans leur formulation.

Diverses approches considérant des informations du domaine spatial telles que les histogrammes de valeurs ont été proposées pour caractériser de telles textures, mélangeant analyses spectrale et spatiale. Cette approche est par exemple utilisée par Dischler et Ghazanfarpour [DG97] pour l'analyse et la reproduction de textures géométriques en se basant sur des **DFT** et des histogrammes de valeurs d'une image. Cette approche est reprise par Gilet et Dischler [GD10], comparant des histogrammes calculés non pas sur les valeurs d'un motif, mais sur la réponse locale (autour d'un pixel) d'un filtre évalué sur ses fréquences, et unifiant les informations spatiales et spectrales du motif. La réponse du filtre à une fréquence donnée, au voisinage d'un pixel donné, est obtenue par une transformation de Fourier fenêtrée. Les différents histogrammes sont obtenus par application du filtre sur différentes résolutions de l'image.

Les bruit de convolution éparse [Lew89] et bruit de spot [vW91] ouvrent également la voix à un possible bruit procédural orienté sur une définition spatiale à l'aide d'un noyau adapté. Lagae *et al.* ont proposé un bruit de convolution procédural en utilisant le noyau de Gabor pour permettre une édition spectrale d'un bruit, Une approche similaire pourrait être envisagée pour l'édition spatiale, en définissant un noyau à partir de primitives simples. Idéalement, cette primitive doit pouvoir facilement s'adapter à la dimension voulue du motif, permettre une approche spectrale de l'édition et l'extraction automatique de paramètres par un processus d'analyse.

Pour créer un tel bruit procédural, nous avons choisi à travers cette thèse d'utiliser une gaussienne elliptique comme primitive pour la définition spatiale d'un noyau de convolution. Une telle primitive est utilisée dans de nombreux domaines pour la visualisation scientifique, notamment dans les algorithmes de rendu par projection et rastérisation de points (voir section 1.4.2), et pour la modélisation de surfaces et de volumes [JBL*06]. Cette primitive offre également la possibilité d'extraire un paramétrage de noyau par analyse d'exemple, en utilisant des méthodes de décomposition d'images en ellipses similairement à celle utilisée par Galerne *et al.* [GLLD12] pour la décomposition des bandes de fréquences.

1.4 Le rendu volumétrique temps réel sur GPU

Pour la visualisation de scène, il est courant de considérer la scène comme un simple ensemble de surfaces situé dans un espace vide. Mais cette supposition ne permet pas de reproduire visuellement certains phénomènes naturels de manière réaliste : l'apparence des nuages, du brouillard et même de la peau est liée à l'absorption et à la diffusion de la lumière traversant un milieu participant contenu dans un volume. Pour le rendu de ce volume, ce milieu est souvent considéré comme un ensemble de particules microscopiques pouvant absorber et émettre de l'énergie afin de simplifier l'évaluation du parcours de la lumière dans celui-ci. Le rendu volumétrique est depuis longtemps utilisé pour la visualisation de données scientifiques acquises par le biais d'un scanner ou d'un microscope, mais également dans la production cinématographique et vidéoludique pour le rendu physiquement réaliste.

Pour le rendu volumétrique temps réel, plusieurs méthodes permettent d'évaluer la contribution du volume dans l'image, ce volume pouvant être présenté en mémoire sous différentes formes (voir section 1.1.3). Similairement aux méthodes de rendu de surface, ces différentes méthodes de rendu se séparent en deux catégories : les méthodes d'ordre image, évaluant par pixel la contribution du volume à l'image, et les méthodes d'ordre objet, évaluant cette même contribution par élément de volume.

Nous rappelons ci-après les notions fondamentales du transport de la lumière dans un volume, ainsi que les différentes méthodes de rendu volumétrique temps réel pouvant être implantées sur carte graphique. Nous rappelons en premier lieu le modèle émission-absorption, servant de base mathématique à de nombreuses approximations utilisées dans les méthodes de rendu volumétrique. Nous présentons ensuite les méthodes d'ordre objet les plus utilisées et les possibles implantation du lancer de rayon sur carte graphique pour le rendu volumétrique.

1.4.1 Le transport de la lumière dans un volume

Nous représentons ici brièvement les principales dérivations mathématiques décrivant le transport de la lumière dans un média participant tel que décrit par Pharr et Humpreys [PH10] prenant comme exemple la traversée d'un rayon de lumière dans un cylindre différentiel rempli de particules, et par Crassin [Cra11]. En particulier, nous rappelons les principes du modèle mathématique « absorption-émission-diffusion » décrivant le comportement de la lumière traversant un nuage de particules dans un espace 3D. Selon ce modèle, ce nuage de particules modifie la distribution de la luminance dans ce volume de trois manières différentes :

- Les particules peuvent absorber une partie de l'énergie lumineuse, celle-ci se transformant en une autre forme (par exemple de la chaleur), résultant en une diminution de la luminance.
- Certaines de ces particules peuvent également réémettre une partie de cette énergie sous forme lumineuse.
- Les rayons de lumière venant d'une direction vont également être réfléchis dans différentes directions pour chaque particule, créant une diffusion de la lumière dans le volume.

Ces interactions peuvent être homogènes ou hétérogènes à l'intérieur du volume, c'est-à-dire être identiques dans l'intégralité du volume ou varier dans différentes régions de celui-ci.

L'absorption est décrite par un facteur σ_a, représentant la densité de probabilité par unité minimale de distance pour laquelle la lumière est absorbée par le volume, exprimé en m^{-1}. Ce facteur d'absorption est généralement dépendant de la position $\mathbf{p}$ dans le volume, mais une direction ω est également prise en compte dans sa formulation exacte. Reprenant l'exemple du cylindre différentiel de Pharr et Humpreys, pour une luminance incidente $L_i(\mathbf{p}, -\omega)$ et une luminance sortante $L_o(\mathbf{p}, \omega)$, le changement de luminance le long du rayon est décrit par l'équation différentielle suivante :

$$L_o(\mathbf{p}, \omega) - L_i(\mathbf{p}, -\omega) = \mathrm{d}L_o(\mathbf{p}, \omega) = -\sigma_a(\mathbf{p}, \omega) L_i(\mathbf{p}, -\omega)\mathrm{d}t$$

Pour une distance d parcourue par le rayon dans le volume, l'absorption totale est alors donnée par :

$$\mathrm{e}^{-\int_0^d \sigma_a(p+t\omega,\omega)dt}$$

À l'inverse de l'absorption, l'émission considère qu'une luminance supplémentaire est renvoyée par les particules du volume. Cette luminance peut être le résultat d'une réaction chimique, thermique ou même nucléaire transformant de l'énergie en lumière. Considérant cette émission comme indépendante de la luminance incidente, le changement de la luminance sortante selon cette émission est donné par l'équation différentielle :

$$\mathrm{d}L_o(\mathbf{p}, \omega) = L_e(\mathbf{p}, \omega)\mathrm{d}t$$

Dans un milieu participant, la lumière tend à se diffuser dans le milieu, celle-ci se réfléchissant sur les particules constituant le volume. Cette diffusion se décompose en deux effets sur la luminance totale d'un rayon traversant une unité de volume :

- Une part du rayon entrant dans ce volume est diffusée dans d'autres directions, provoquant une diminution de la luminance sortante.
- La luminance d'autres rayons peut avoir été réfléchie par les particules du volume unitaire dans la direction du rayon évalué, provoquant une augmentation de la luminance.

Ces deux effets sont respectivement appelés diffusion sortante et diffusion entrante. La diffusion est plus généralement donnée par un coefficient σ_s définissant la probabilité qu'un événement de diffusion se produise par unité de distance. La modification de la luminance sortante selon la diffusion sortante est décrite par l'équation différentielle :

$$\mathrm{d}L_o(\mathbf{p},\omega) = -\sigma_s(\mathbf{p},\omega)L_i(\mathbf{p},-\omega)$$

Ce facteur de diffusion combiné avec le facteur d'absorption nous donne ainsi le facteur d'atténuation (aussi appelé facteur d'extinction) σ_t définissant la diminution totale de luminance par unité de distance :

$$\sigma_t(\mathbf{p},\omega) = \sigma_s(\mathbf{p},\omega) + \sigma_a(\mathbf{p},\omega)$$

Ce dernier coefficient permet finalement de définir la transmittance du volume, c'est-à-dire la fraction de luminance transmise depuis un point $\mathbf{p}$ jusqu'à un point $\mathbf{p}'$:

$$T_r(\mathbf{p}\to\mathbf{p}') = \mathrm{e}^{-\int_0^{||\mathbf{p}'-\mathbf{p}||}\sigma_t(p+t\omega,\omega)dt} \qquad \textbf{avec} \qquad \omega = \frac{\mathbf{p}'-\mathbf{p}}{||\mathbf{p}'-\mathbf{p}||} \tag{1.15}$$

L'exposant négatif de cette exponentielle est également appelé *épaisseur optique*, représenté par le symbole τ :

$$\tau_t(\mathbf{p}\to\mathbf{p}') = \int_0^{||\mathbf{p}'-\mathbf{p}||}\sigma_t(\mathbf{p}+t\omega,\omega)\mathrm{d}t \tag{1.16}$$

Une propriété importante de cette transmittance est qu'elle est multiplicative le long du rayon :

$$T_r(\mathbf{p}\to\mathbf{p}'') = T_r(\mathbf{p}\to\mathbf{p}')T_r(\mathbf{p}'\to\mathbf{p}'') \tag{1.17}$$

Dans un milieu homogène, le facteur d'atténuation σ_t est considéré constant, et permet de simplifier l'équation 1.15 :

$$T_r(\mathbf{p}\to\mathbf{p}') = \mathrm{e}^{-\sigma_t||\mathbf{p}'-\mathbf{p}||} \tag{1.18}$$

La modification de la luminance totale en prenant en compte la diffusion entrante est plus généralement donnée par le terme de source S, qui définit la luminance totale émise par unité de distance :

$$\mathrm{d}L_o(\mathbf{p},\omega) = S(\mathbf{p},\omega)\mathrm{d}t$$

Ce terme S décrit cette luminance selon l'équation suivante :

$$S(\mathbf{p},\omega) = L_e(\mathbf{p},\omega) + \sigma_s \int_{\mathbf{S}^2} p(\mathbf{p},-\omega' \rightarrow \omega) L_i(\mathbf{p},\omega') \mathrm{d}\omega' \tag{1.19}$$

La diffusion entrante est obtenue par la multiplication du coefficient de diffusion σ_s à l'intégrale sphérique des luminances incidentes modifiées par une fonction de phase p. Cette dernière décrit la distribution angulaire de la diffusion d'énergie en un point, c'est-à-dire la quantité de luminance redistribuée dans chaque direction.

Le terme source et le facteur d'atténuation permettent finalement de définir l'équation différentiel du transfert de luminance à travers le volume pour une distance unitaire :

$$\frac{\partial}{\partial t} L_o(\mathbf{p},\omega) = -\sigma_t(\mathbf{p},\omega) L_i(\mathbf{p},-\omega) + S(\mathbf{p},\omega) \tag{1.20}$$

L'équation de transfert sous sa forme intégrale se présente principalement sous deux formulations. Dans le cas où le rayon n'intersecte aucune surface, la luminance en un point $\mathbf{p}$ selon une direction ω peut être obtenue par l'accumulation de la luminance de l'intégralité des points du rayon partant du point $\mathbf{p}$ dans la direction ω :

$$L_i(\mathbf{p},\omega) = \int_0^{\infty} T_r(\mathbf{p}' \rightarrow \mathbf{p}) S(\mathbf{p}',-\omega) \mathrm{d}t' \tag{1.21}$$

Où $\mathbf{p}' = \mathbf{p} + t\omega$ est un point quelconque sur le rayon. La luminance accumulée en chaque point du rayon décroît par l'effet de la fonction de transmittance. Dans le cas plus général, les rayons évalués ne sont pas infinis et peuvent intersecter des surfaces, modifiant la luminance sortante de la surface et empêchant les points situés le long du rayon après cette intersection de contribuer à la luminance située à l'origine du rayon. L'équation de transfert est dans ce cas :

$$L_i(\mathbf{p},\omega) = T_r(\mathbf{p_0} \rightarrow \mathbf{p}) L_o(\mathbf{p_0},-\omega) + \int_0^{t} T_r(\mathbf{p}' \rightarrow \mathbf{p}) S(\mathbf{p}',-\omega) \mathrm{d}t' \tag{1.22}$$

Où $\mathbf{p_0} = \mathbf{p} + t\omega$ est le point d'origine du rayon (par exemple une surface) et $\mathbf{p}' = \mathbf{p} + t'\omega$ est un point quelconque le long du rayon. Dans cette dernière équation, le premier terme contenant L_o correspond à la luminance émise par la surface le long du rayon évalué. Le second terme évalue l'atténuation et l'émissivité du volume pour la direction du rayon évalué.

De par l'intégrale du rayon mais plus particulièrement de par l'intégrale sphérique du S (éq. 1.19), cette équation est difficilement évaluable en temps réel sans introduire certaines approximations ou certaines hypothèses de modélisation permettant de simplifier le calcul. Une approximation courante de ce modèle consiste à négliger le phénomène de diffusion, c'est-à-dire à considérer σ_s nul. On parle dans ce cas du modèle « émission-absorption ». Pour les équations suivantes, nous considérons également la luminance sortante comme étant indépendante d'une direction. L'équation de transfert de la luminance

en un point $\mathbf{p}$ depuis un point $\mathbf{p_0}$ est alors approximée par l'équation suivante :

$$L(\mathbf{p}) = \mathrm{e}^{-\tau_a(\mathbf{p_0}\to\mathbf{p})}L_0 + \int_0^{||\mathbf{p_0}-\mathbf{p}||} \mathrm{e}^{-\tau_a(\mathbf{p'}\to\mathbf{p})}L_e(\mathbf{p'})\mathrm{d}t \quad \textbf{avec} \quad \mathbf{p'} = \mathbf{p} + t\frac{\mathbf{p_0}-\mathbf{p}}{||\mathbf{p_0}-\mathbf{p}||} \tag{1.23}$$

L_0 est la luminance au point d'origine $\mathbf{p_0}$ et τ_a fait référence à l'épaisseur optique sans diffusion. Dans le cas général, cette équation ne peut pas être résolue de manière analytique, mais elle peut être approximée par un découpage en N sous-parties. Par extension de l'équation précédente, la luminance en un point $\mathbf{p}_i$ peut être reformulée en :

$$L(\mathbf{p}_i) = \mathrm{e}^{-\tau_a(\mathbf{p}_{i-1}\to\mathbf{p}_i)}L_{i-1} + \int_0^{||\mathbf{p}_{i-1}-\mathbf{p}_i||} \mathrm{e}^{-\tau_a(\mathbf{p'}\to\mathbf{p}_i)}L_e(\mathbf{p'})\mathrm{d}t \tag{1.24}$$

Dans cette équation, la transmittance $T_i = T_r(\mathbf{p}_{i-1} \to \mathbf{p}_i) = \mathrm{e}^{-\tau_a(\mathbf{p}_{i-1}\to\mathbf{p}_i)}$ représente la transparence de l'intervalle i, et $C_i = \int_0^{||\mathbf{p}_{i-1}-\mathbf{p}_i||} T_r(\mathbf{p'} \to \mathbf{p}_i)L_e(\mathbf{p'})\mathrm{d}t$ est la luminance contribuant à l'intervalle i, C_0 étant égale à L_0. La fonction de transfert peut alors être décrite par la forme discrète suivante :

$$L(\mathbf{p}) = \sum_{i=0}^{N}\left(C_i \prod_{j=i+1}^{N} T_j\right) \tag{1.25}$$

Pour simplifier l'évaluation, il est courant de considérer le volume comme étant un milieu homogène par intervalles, ce qui permet d'approximer la transparence d'un intervalle de taille Δ_i similairement à l'équation 1.18 :

$$T_i \approx \mathrm{e}^{-\sigma_a(\mathbf{p}_i)\Delta_i}$$

La composante C_i est souvent assimilée à la contribution d'une couleur, obtenue par l'approximation :

$$C_i \approx L_e(\mathbf{p}_i)\Delta_i$$

L'évaluation de l'équation 1.25 peut être réalisée de manière incrémentale, par accumulation successive de la luminance et de la transparence. Cette accumulation est plus couramment réalisée en prenant comme point de départ l'œil, jusqu'à atteindre une valeur de transparence nulle [KW03]. L'accumulation de la luminance et de la transparence se traduit dans ce cas par les opérations récursives suivantes :

$$\begin{aligned} L(\mathbf{p}_i) &= L(\mathbf{p}_{i+1}) + T_{i+1}C_i \\ T_i &= T_iT_{i+1} \end{aligned} \tag{1.26}$$

Cette évaluation permet par la suite de recomposer une image en utilisant l'opération de mélange suivante en définissant l'opacité $\alpha = 1 - T$:

$$\begin{aligned} C_{dst} &= C_{dst} + (1-\alpha_{dst})C_{src} \\ \alpha_{dst} &= \alpha_{dst} + (1-\alpha_{dst})\alpha_{src} \end{aligned} \tag{1.27}$$

1.4.2 Les méthodes de rendu d'ordre objet

Tout comme le rendu temps réel de surface sur carte graphique, le rendu volumétrique peut être réalisé en utilisant le pipeline graphique pour calculer la contribution des différents voxels à l'image. Pour le rendu de nuage de points dans un volume plus particulièrement, il est possible d'utiliser une approche par projection directe des voxels, ou bien de réaliser un échantillonnage du volume par projection de coupes géométriques.

La projection de voxels

Parmi les méthodes les plus anciennes pour le rendu temps réel de volume, le *volume splatting* [Wes90, KB04] se base sur la projection directe des voxels dans l'espace écran : la contribution de chaque voxel est évaluée par pixel en projetant pour chaque voxel un point associé à un noyau de reconstruction (la littérature parle de *splat* dans ce cas). L'utilisation d'une gaussienne elliptique telle que celle présentée par Zwicker *et al.* [ZPvBG01] comme noyau de reconstruction permet un filtrage analytique des éléments de volume selon la distance de vue. Cette gaussienne elliptique centrée en un point $\mathbf{p}$ et dont l'isocontour est donné par une matrice de variance $\mathbf{V}$ est définie par l'équation suivante :

$$G_{\mathbf{V}}(\mathbf{x}-\mathbf{p}) = \frac{|\mathbf{V}^{-1}|^{\frac{1}{2}}}{2\pi} \mathrm{e}^{-\frac{1}{2}(\mathbf{x}-\mathbf{p})^T \mathbf{V}^{-1}(\mathbf{x}-\mathbf{p})} \tag{1.28}$$

Où $\mathbf{x}$ et $\mathbf{p}$ sont des vecteurs colonnes et $\mathbf{V}$ est une matrice symétrique de dimension 3×3. Le filtre de reconstruction en espace écran d'un noyau k évalué en un pixel $\hat{\mathbf{x}}$ est obtenu par la convolution d'un filtre passe-bas avec la gaussienne elliptique projetée :

$$\rho_k(\hat{\mathbf{x}}) = \frac{1}{|\mathbf{J}^{-1}||\mathbf{W}^{-1}|} G_{\hat{\mathbf{V}}_k + \mathbf{V}_h}(\hat{\mathbf{x}} - \hat{\mathbf{x}}_k) \tag{1.29}$$

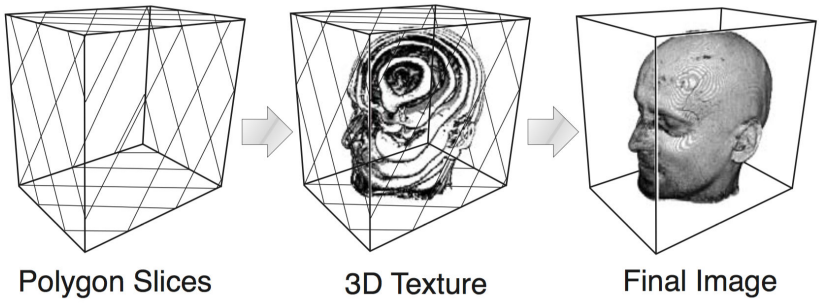
Où $\hat{\mathbf{V}}_k$ est la matrice de variance $\mathbf{V}_k$ du noyau k projeté dans l'espace écran, ramenée à une dimension 2×2 (la dernière ligne et la dernière colonne de la matrice sont supprimées). $\mathbf{V}_h$ est la matrice de variance du filtre passe-bas (typiquement une matrice identité), $\mathbf{J}$ est la jacobienne du pixel, $\mathbf{W}$ est la matrice de transformation de l'espace objet vers l'espace caméra et $\hat{\mathbf{x}}_k$ est la position du noyau k projetée dans l'espace écran.

Le rendu par projection permet d'estimer rapidement l'empreinte du volume dans l'image par accumulation des contributions des différents voxels après leur projection dans l'espace écran. Mais les différentes méthodes de rendu par projection partent de l'hypothèse que les noyaux de reconstruction sont projetés du plus proche de la caméra au plus lointain, ce qui peut impliquer un tri coûteux lors du rendu. Une projection non ordonnée provoque l'apparition d'artefacts de coloration dans le cas des volumes composés de différentes matières, certains voxels contribuant aux rayons de vue là où un échantillonnage incrémental du rayon aurait déjà atteint son opacité maximale. Cet ordonnancement peut être approximé à moindre coût par des méthodes dites d'*Order Independent Transparency* (**OIT**), recalculant la contribution d'une partie du volume en fonction de sa profondeur dans l'image. Le *depth peeling* [Eve01] permet par exemple d'approximer l'évaluation incrémentale par rendu par projection : à chaque passe de rendu, les voxels

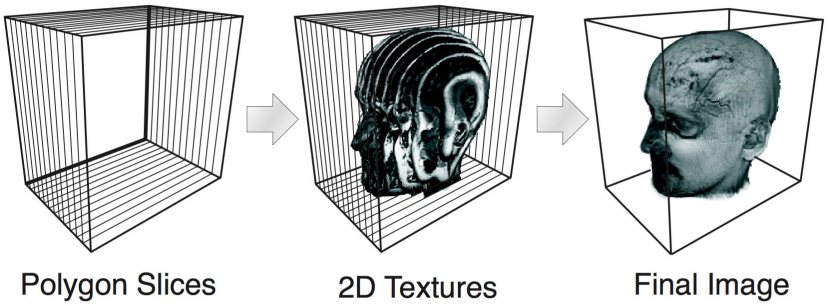
compris entre deux plans de coupes parallèles à l'écran se déplaçant le long du rayon de vue sont évalués, et leur contribution à l'image est mélangée au résultat des passes de rendu précédentes. Pour une meilleure approximation, d'autres méthodes profitent des améliorations du matériel graphique et implantent sur la carte graphique un tampon de transparence (ou *A-Buffer*) [MET*14] ou bien encore des listes chaînées de fragments par pixel [SV14, Wym16]. Plus de détails sur ces différentes méthodes de gestion de la transparence sont donnés dans l'article de Maule *et al.* [MCTB12].

La projection de coupes

Les méthodes de rendu par projection de coupes (ou *slicing*) [EHK*04], contrairement au rendu par projection de voxels, réalisent un échantillonnage du volume en générant des coupes géométriques dans celui-ci. L'idée générale de ces méthodes est de rastériser des polygones définissant des coupes du volume 3D pour échantillonner les données volumiques stockées sous la forme d'un ensemble de textures 2D ou d'une texture 3D. Ces coupes peuvent être alignées selon la direction de la caméra [WE98], ou bien alignées selon différents axes de l'objet [RSEB*00] (figure 1.32).



(a) Tranches alignées selon la vue



(b) Tranches alignées selon un axe de l'objet

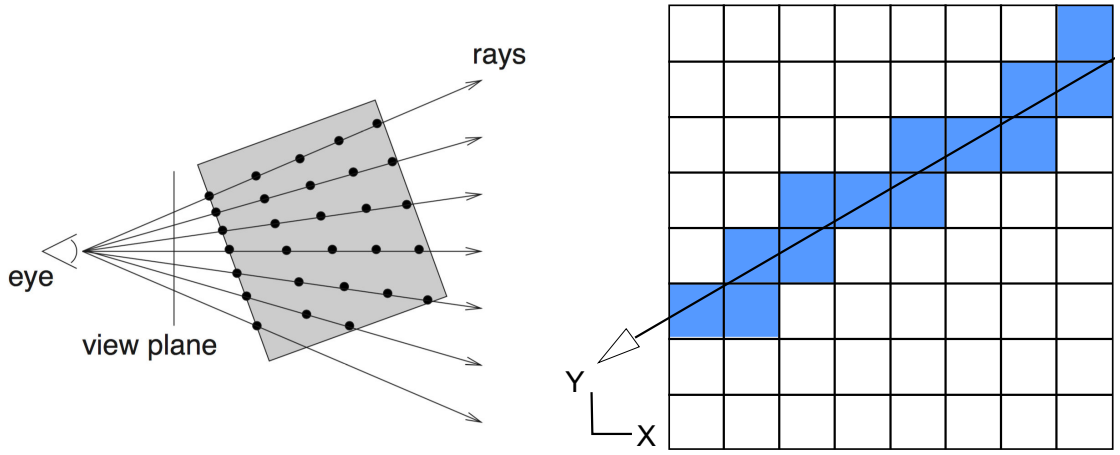
FIGURE 1.32 – Rendu volumétrique en coupes alignées [EHK*04].

Les méthodes de rendu par projection de coupes sont préférées à la projection de voxels dans de nombreux cas où la qualité prime sur la vitesse : les coupes du volume générées de façon ordonnée selon la direction de vue permettent d'évaluer efficacement l'équation 1.25. Ce type de méthode est largement utilisé pour le rendu temps réel des surcouche et textures volumétriques présentés dans la section 1.1.3.

1.4.3 Le lancer de rayon sur GPU

La souplesse grandissante des *fragment shaders*, les interfaces de programmation pour le calcul parallèle sur carte graphique (i.e. CUDA / OpenCL) ou encore les *compute shaders*, permettent d'implanter différentes méthodes d'évaluation d'ordre image, c'est-à-dire dont l'évaluation part du pixel. Les méthodes de lancer de rayon par pixel sur GPU se sont largement démocratisées depuis que le pipeline graphique peut être programmés. Elles sont utilisées pour le rendu temps réel haute qualité de nombreux effets tels que du brouillard volumique [Wro14], ou encore pour des calculs d'illuminations [CNS*11]. Leur principe est simple : Chaque pixel produit un rayon, le point de départ de ce rayon étant obtenu par rastérisation d'une surface ou, dans le cas des volumes, de la boite englobante des données volumétriques.

Le lancer de rayon permet une évaluation de la contribution du volume à l'aide des méthodes de *ray-marching* : la contribution du volume au pixel est obtenue par un échantillonnage pas à pas des données volumétriques situées à différents points du rayon (figure 1.33). L'évaluation peut être interrompue lorsqu'un certain nombre d'échantillons a été évalué, ou bien si l'opacité maximale ou la sortie du volume est atteinte. L'implantation de cette méthode par Kruger et Westermann [KW03] est originellement une approche multi-passes de rendu permettant l'arrêt de l'évaluation. Mais des implantations n'effectuant qu'une unique passe de rendu ont été réalisées depuis que le pipeline graphique programmable gère plus efficacement les branchements dans les algorithmes [SSKE05].



(a) Échantillonnage à intervalle fixe (b) Échantillonnage d'une grille régulière

FIGURE 1.33 – Lancer de rayon dans un volume [EHK*04]

Un des avantages du lancer de rayon est son adaptabilité à différentes représentations. Pour un volume représenté par une grille régulière de voxels, l'échantillonnage du rayon peut être réaliser de manière incrémentale en utilisant un algorithme de Bresenham pour parcourir les voxels traversés par le rayon (figure 1.33.b). L'utilisation d'une telle grille régulière requiert cependant une grande quantité de mémoire pour le stockage de volume haute résolution et les voxels transparents augmentent la quantité d'échantillons évalués avant l'arrêt de l'évaluation. Les octrees [RV06, JM16] permettent une évaluation plus efficaces pour le rendu temps réel. Cette structure arborescente permet de réduire le

nombre d'échantillons évalués en passant plus rapidement les ensembles vides dans le volume. Dans le cas des textures volumiques procédurales, pour lesquelles les données de volume ne sont connues qu'à l'évaluation, ce même échantillonnage est simplement réalisé à intervalle régulier. La taille de cet intervalle devient un paramètre de la méthode de rendu (figure 1.33.a).

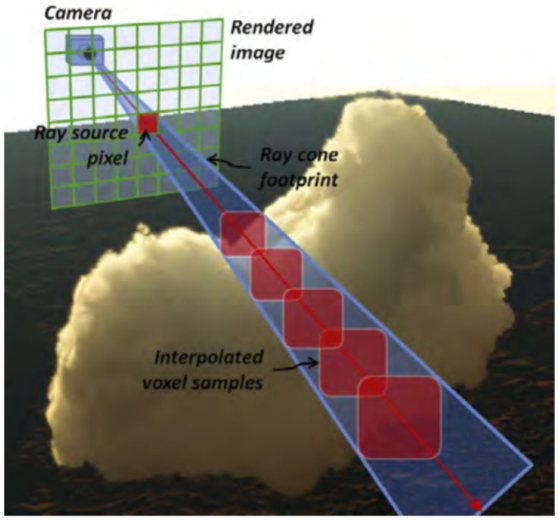


FIGURE 1.34 – Lancer de cone dans un volume [Cra11]

L'échantillonnage pas à pas du rayon permet un rendu volumétrique temps réel efficace, mais cette approche considère le centre du pixel comme un unique point de départ d'un rayon de vue. Or dans une image obtenue par une projection en perspective, la déprojection de ce même pixel dans le volume forme une pyramide. Une approximation possible de cette pyramide est de considérer un pixel comme le point de départ d'un cône de rayon de vue [Ama84]. Crassin *et al.* [CNS*11] réalisent un rendu par lancer de cône (figure 1.34) grâce aux possibilités de filtrage des textures 3D, en créant notamment un octree à partir de différents niveaux de *MIP-Maps*. Un échantillonnage pas à pas est toujours utilisé, mais le cône du rayon est approximé en accédant à différents niveaux de l'octree lors de l'échantillonnage du rayon.

1.4.4 Synthèse

L'amélioration du matériel graphique permet d'utiliser en temps réel de plus en plus de techniques de rendu utilisées auparavant pour la production de films d'animation haute qualité. Les méthodes basées sur le lancer de rayon peuvent exploiter très efficacement la puissance de calcul parallèle brute des cartes graphiques grâce aux *fragment shaders* et aux *compute shaders* des interfaces de programmation graphique (Direct3D 11+/Opengl 4.3+) ou de celles de programmation parallèle générique (CUDA/OpenCL). Ces méthodes permettent d'évaluer plus efficacement par pixel des effets complexes d'illumination ou encore le transport de la lumière dans un volume. Pour le rendu volumétrique temps réel, comparativement aux méthodes d'ordre objet, l'échantillonnage pas à pas d'un rayon par pixel compromet moins la qualité du résultat et évalue ou génère moins d'échantillons non contributifs. La contrepartie assumée par cette approche est que

la majorité de la complexité de rendu est déportée sur les unités de calcul par pixel. Les méthodes d'ordre objet peuvent faire appel à d'autres unités de calcul spécifiques pour accélérer certaines opérations, telles que la création des échantillons du volume réalisée par les unités de rastérisation dans le cas du rendu par projection de coupes. Un problème peut également se poser pour l'évaluation de textures volumiques procédurales : une texture procédurale incluant des variations hautes fréquences (ou, dans notre cas d'étude, des éléments infinitésimaux tels que des brins) peut nécessiter un pas d'échantillonnage très petit afin de ne manquer aucun échantillon contributif important.

La projection de voxels est souvent considérée comme la méthode d'ordre objet la plus rapide pour l'évaluation temps réel par la carte graphique : l'évaluation de la contribution du volume dans l'image se résume à une convolution dans l'espace écran. L'évaluation des différents éléments de volume peut être réalisée par projection et rastérisation de simples quadrilatères sur lesquels sont plaqués les noyaux de reconstruction. Mais cette approximation peut créer des artefacts de recouvrement entre les voxels et des couleurs incorrectes. Malgré ces artefacts, la performance du rendu par projection de voxels présente un intérêt non négligeable, en particulier pour le rendu de voxels ne se recouvrant pas ou dans le cas d'éléments volumiques plats, les méthodes de rendu par projection de coupes ou par échantillonnage pas à pas de rayons pouvant dans les autres cas être utilisées. Parmi les améliorations du matériel graphique pouvant profiter à cette méthode de rendu, l'instanciation de géométrie permet notamment d'accélérer le rendu des supports d'évaluation des voxels (ou *splats*). Une approximation rapide de l'ordonnancement peut être réalisée par voxel en utilisant une approche similaire à celle proposée par McGuire et Bavoil [MB13] : chaque *splat* peut être pondéré selon sa profondeur en utilisant une fonction à forte décroissance telle que celles proposées par McGuire et Bavoil. Des artefacts de couleur peuvent cependant toujours apparaître dus au mélange des contributions, l'artefact de couleur s'intensifiant lorsque la différence de poids entre deux éléments de volume est trop faible.

Nous étudions à travers cette thèse la possibilité de rendre en temps réel une texture procédurale volumique représentant des brins. Pour l'évaluation de cette texture, partant des observations précédentes, nous nous intéressons aux possibilités d'adapter les méthodes d'échantillonnage de rayons et de projection de volumes pour le rendu d'éléments volumiques très fins.

Chapitre 2

Synthèse de Texture Procédurale par Bruit de Spot Localement Contrôlé

Sommaire

Chapitre 2

Synthèse de Texture Procédurale par Bruit de Spot Localement Contrôlé

Les résultats présentés dans ce chapitre ont fait l'objet d'une présentation à la conférence internationale *Winter School of Computer Gaphics* 2016 [PGDG16].

2.1 Introduction

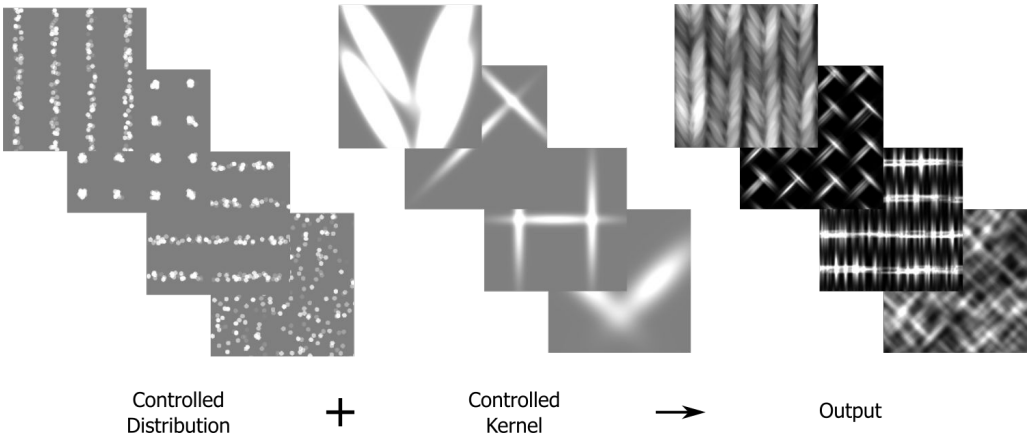


FIGURE 2.1 – Résumé du modèle du bruit de spot localement contrôlé.

Les signaux aléatoires ont été largement utilisés pour la création de textures procédurales depuis le motif de marbre de Perlin [Per85]. Les textures procédurales basées sur les bruits procéduraux héritent de certaines propriétés de ceux-ci, les plus intéressantes étant :

- Aucune répétition n'est visible.
- Le motif produit est continu sur l'espace d'évaluation.
- Il peut être évalué par pixel pendant le rendu.
- Une unique modèle de texture peut produire de nombreux motifs différents par simple modification des paramètres.

Ces différents avantages ont conduit à de nombreux travaux sur l'utilisation des bruits procéduraux pour la création de textures procédurales. Une grande variété de motifs peut être produite par des fonctions de bruit en définissant un spectre de puissance particulier

mais créer un motif par édition du spectre de puissance du bruit est une tâche complexe : la corrélation entre le spectre de puissance et l'apparence est difficilement déductible sans un long apprentissage. Une approche mieux adaptée aux artistes consiste à extraire les paramètres du bruit à partir d'un exemple du motif (section 1.3.4).

Les bruits procéduraux paramétrables par analyse d'exemple ont pour objectif de générer un motif de bruit reproduisant le spectre de puissance d'une image passée en entrée et subissant une analyse spectrale. La variété visuelle du résultat est obtenue par l'utilisation de phases aléatoires. Cependant, des informations de structure peuvent être contenues dans le spectre de phase de l'analyse spectrale [OL81]. Le bruit local à phase aléatoire [GSV*14] résout le problème de conservation des éléments structurels en fixant les phases et magnitudes dans certaines zones du spectre fréquentiel, approximant la composante structurelle de l'exemple passé en entrée. Mais cette approche pose deux problèmes. Premièrement, les informations de phases correspondant à ces éléments structurels doivent être stockées. Deuxièmement, un nombre limité de fréquences fixées étant utilisé, une fonction de turbulence [Per85] et des décalages aléatoires doivent être utilisés pour casser la périodicité des éléments structurels produits.

Nous présentons ci-après une formulation alternative du bruit local à phase aléatoire basée sur un bruit de spot procédural pouvant être localement défini et contrôlé. Nous nous concentrons sur les éléments structurels pouvant être définis par un noyau structuré répétitif, tels que la forme des coutures et les micro-variations présentes dans une texture de tissu. Ce type de noyau peut être créé par une disposition de composants élémentaires de ces éléments structurels, tels que les fils dans les coutures de cette texture de tissu. Cette formulation conserve les avantages du bruit local à phase aléatoire : les détails aléatoires et les éléments structurels peuvent être générés en temps réel par pixel. Dans le cas de motifs composés d'éléments structurels simples, la configuration des noyaux peut être extraite par un processus d'analyse automatique. Notre formulation a deux avantages par rapport à celle du bruit local à phase aléatoire :

- La somme de cosinus représentant la composante structurelle du bruit local à phase aléatoire est remplacée par une somme de quelques noyaux gaussiens. Cette représentation compacte réduit le coût d'évaluation des structures locales et permet d'obtenir un bruit procédural aux performances similaires mais profitant d'un contrôle plus fin sur l'apparence visuelle finale.
- La distribution des différents éléments structurels peut être modifiée et fait partie de la définition du modèle. Le nombre de motifs pouvant être produit par ce modèle s'en trouve augmenté, ces motifs pouvant varier de quasi régulier à complètement aléatoire tout en comportant toujours des éléments structurels. La distribution des bruits locaux se basant sur un processus aléatoire permet d'éviter les répétitions et la périodicité des motifs semi-réguliers.

Le contrôle simultané sur la distribution et sur l'aspect du noyau de notre formulation permet une édition interactive de motifs. Une comparaison des formulations de bruits locaux est présentée dans la figure 2.2. Les possibilités offertes par notre nouvelle formulation sont présentées à travers différents exemples de motifs dans la figure 2.4.

2.2 Modèle du bruit

Nous présentons maintenant notre formulation alternative du bruit local à phase aléatoire basée sur le bruit de spot. Pour rappel, le bruit local à phase aléatoire est initialement défini par l'équation suivante :

$$n(x) = \sum_{i=1}^{I} w\left(\frac{||x - x_i||}{\Delta}\right) \sum_{j=1}^{J} A_{i,j} \cos(2\pi f_{i,j} \cdot x + \rho_{i,j}) \tag{2.1}$$

Ce bruit est un mélange de $J \times I$ bruits locaux basés sur des cosinus, utilisant des phases aléatoires et étant fenêtrés sur une grille régulière (x_i est la position correspondant à la strate i du spectre fréquentiel sur la grille régulière). La partie aléatoire du motif est obtenue par l'utilisation de phases aléatoires, tandis que le spectre est contrôlé par l'échantillonnage des fréquences pour chacun des bruits locaux.

Nous proposons une formulation alternative basée sur un bruit de spot entièrement procédural pour limiter ce nombre de cosinus. Nous introduisons conjointement une nouvelle fonction de distribution procédurale pour contrôler la localité du bruit produit. L'étendue des motifs pouvant être générés par ce nouveau modèle est illustrée par la figure 2.4.

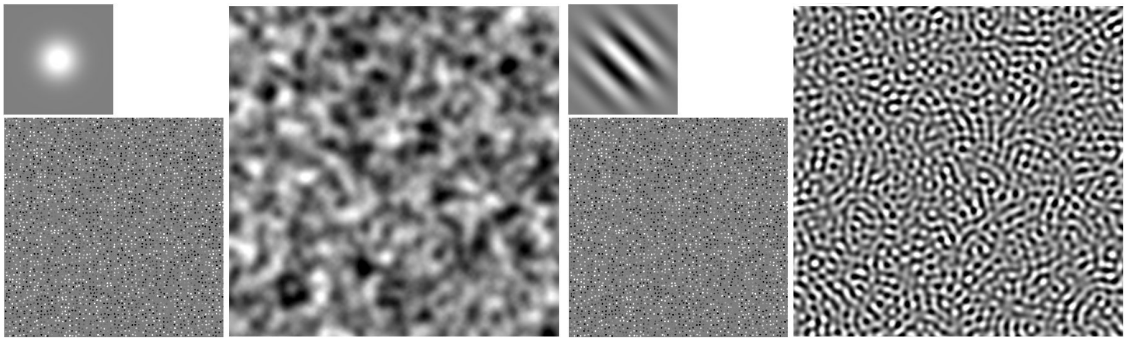
2.2.1 Le bruit de spot procédural

Le bruit de convolution éparse [Lew89] est initialement basé sur une distribution aléatoire d'impulsions convoluée à un noyau gaussien isotrope. Ce noyau, créé par la multiplication d'un échantillon de texture par une enveloppe gaussienne, ne produit que des motifs gaussiens isotropes dus à l'enveloppe gaussienne fixe du noyau. Le noyau de Gabor [LLDD09] peut être utilisé pour améliorer le contrôle spectral, celui-ci unifiant des caractéristiques spatiales et spectrales. Mais le contrôle spatial s'en trouve diminué. Ce noyau ne permet également de produire que des textures gaussiennes, ce qui représente un sous-ensemble restreint de motifs procéduraux. Le bruit de spot [vW91, dLvL97] peut produire un plus grand nombre de motifs, incluant des motifs non gaussiens contenant des éléments structurels, en utilisant un noyau de convolution arbitraire défini spatialement.

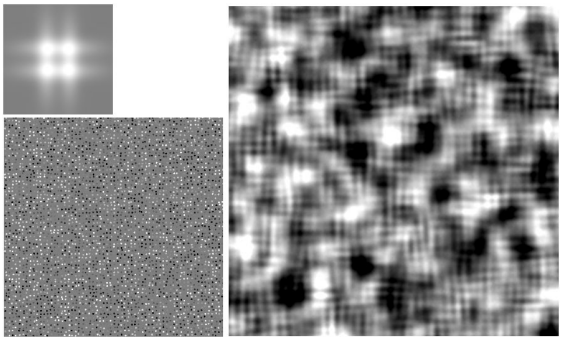
Nous ciblons la reproduction de caractéristiques spatiales ne pouvant être caractérisées par le seul spectre de puissance. Van Wijk [vW91] note que des caractéristiques structurelles présentes dans le noyau lui-même, telles que l'(an)isotropie ou des éléments structurels, sont transposées dans la texture produite par le bruit de spot. Une formulation du bruit de spot est donnée par De Leeuw et Van Liere [dLvL97] par l'équation suivante :

$$n_s(\mathbf{p}) = \sum_{j}^{J} w_j(\mathbf{p}_j)\, k_s\left((\mathbf{p} - \mathbf{p}_j)\, \mathbf{R}_s(\mathbf{p}_j)\, \mathbf{S}_s(\mathbf{p}_j)\right) \tag{2.2}$$

Où $\mathbf{p}_j$ est la position d'une impulsion, et k_s est le noyau de convolution du *Spot* s. L'orientation $\mathbf{R}_s$ et la modification d'échelle $\mathbf{S}_s$ sont liées à des champs de données sous-jacents,



(a) Bruit gaussien (b) Bruit de Gabor



(c) Bruit de spot

FIGURE 2.2 – Exemples de bruits générés par convolution éparse, incluant leur noyau et leur distribution d'impulsions respectifs. Contrairement au bruit gaussien (a) ou au bruit de Gabor (b), le bruit de spot peut produire des éléments structurels semi-réguliers en utilisant un noyau spatial arbitraire.

tels que des champs de vecteurs ou d'intensités. Les positions d'impulsions $\mathbf{p}_j$ sont distribuées de manière uniforme en utilisant une distribution de Poisson et les poids w_j sont tirés de manière équiprobable dans l'intervalle $[-1, 1]$.

2.2.2 Le noyau multi-gaussiennes elliptiques

Pour notre formulation du noyau, nous utilisons une somme de fonctions gaussiennes de dimension N et d'orientations et de tailles arbitraires. Les fonctions gaussiennes, et par extension les noyaux gaussiens, sont couramment utilisées dans la littérature des bruits pour la création de motifs gaussiens. Ces fonctions sont également utilisables pour la modélisation de surfaces et de volumes [JBL*06]. En traitement de l'image, les noyaux gaussiens peuvent être utilisés comme primitive de reconstruction après la décomposition en ondelette de Gabor d'une image [WM03]. Ils sont également utilisés comme primitive de reconstruction dans certains algorithmes de rendu par projection de points (ou *Splatting*, section 1.4.2). Pour la modélisation d'un *spot* utilisé dans le bruit de spot, une grande variété de noyaux peut être créée par combinaison de quelques fonctions gaussiennes. Quelques exemples de noyaux sont présentés dans la figure 2.3.

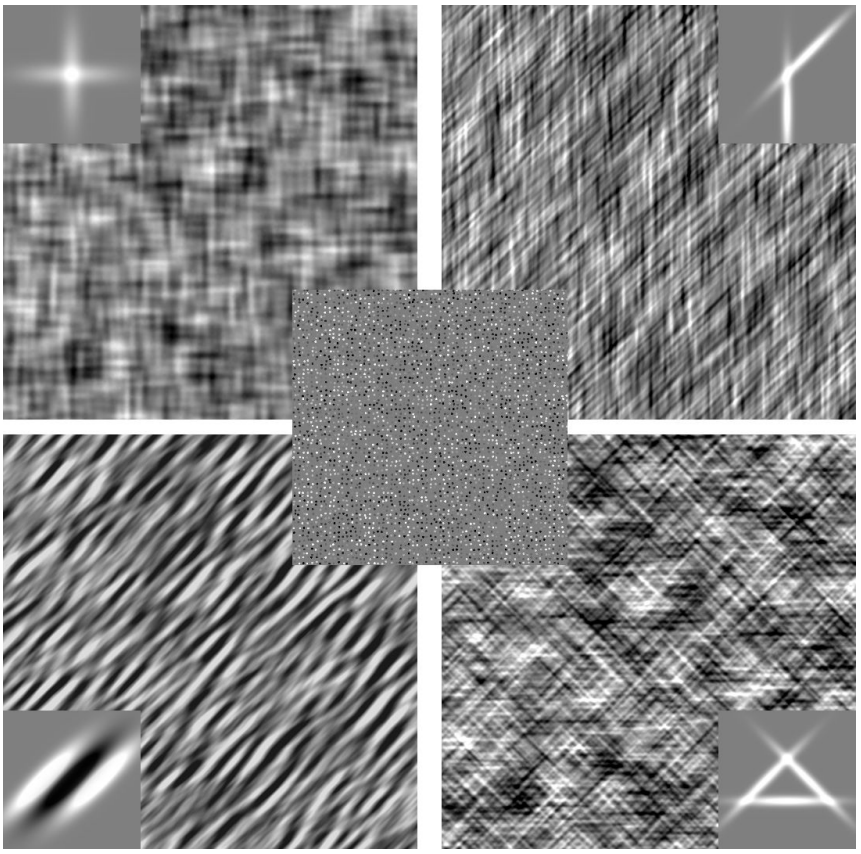


FIGURE 2.3 – Exemples de bruits de spot utilisant différents noyaux composés de fonctions gaussiennes elliptiques.
Chaque gaussienne orientée dans un noyau génère un élément orienté dans le résultat. La tuile centrale illustre la distribution aléatoire d'impulsions utilisée pour chaque bruit. Les résultats des bruits utilisant les *spots* composés de 2 gaussiennes (ligne supérieure) sont générés en ≈ 11.8 millisecondes, et ceux utilisant les *spots* composés de 3 gaussiennes sont générés en ≈ 15.4 millisecondes.

Pour une dimension N, notre noyau est défini par l'équation :

$$k(\mathbf{p}) = \sum_{V_i} g_{V_i} \qquad g_V(\mathbf{p}) = Ae^{-\frac{1}{2}\mathbf{p}^T\mathbf{V}^{-1}\mathbf{p}} \tag{2.3}$$

Où A est la magnitude de la gaussienne, et $\mathbf{V}$ est une matrice de dimension $(N+1)\times(N+1)$, dont l'inverse est définie comme $\mathbf{V}^{-1} = (\mathbf{M}\,\mathbf{R}\,\mathbf{S})^{-T}(\mathbf{M}\,\mathbf{R}\,\mathbf{S})^{-1}$ et $|\mathbf{V}|$ est le déterminant de la matrice. **M**,**R** et **S** sont respectivement des matrices de translation, de rotation et de changement d'échelle. L'isocontour du noyau est défini par l'opération $\mathbf{p}^T\mathbf{V}^{-1}\mathbf{p}$, celle-ci décrivant une surface implicite tant que $\mathbf{V}$ est une matrice semi-positive ($\mathbf{p}$ est un point en dimension $N+1$ dont la dernière composante est égale à 1). Nous présentons dans la figure 2.2.c un exemple où un noyau, dans lequel quatre gaussiennes elliptiques forment un motif de grille, produit une texture comportant une composante structurelle semi-régulière similaire à celle du noyau.

2.2.3 Le bruit de spot procédural localement contrôlé

Les bruits basés sur la convolution éparse considèrent la fenêtre d'évaluation comme étant induite par la formulation du noyau. Dans le cas des fonctions gaussiennes, chaque fonction renvoie une valeur inférieure à une valeur limite de contribution avant d'atteindre la distance d'évaluation maximale du noyau. Considérant cette propriété et l'équation 2.2, la nouvelle formulation de notre modèle de bruit devient :

$$n(\mathbf{p}) = \sum_{i=0}^{I} w_i n_i(\mathbf{p}) \tag{2.4}$$

Où n_i est un bruit local composé d'impulsions aléatoires associées au noyau i. Le coefficient w_i est un facteur de normalisation calculé pour chaque bruit. Notre modèle est une composition de différents bruits de spot, chaque spot servant à modéliser un ensemble spécifique d'éléments structurels du motif.

Les textures générées par ce modèle de bruit contiennent des éléments structurels locaux, similairement aux motifs en forme de grilles de la figure 2.2.c, tout en conservant l'aléatoire introduit par la distribution de Poisson des impulsions. Pour augmenter la variété de motifs pouvant être générés par ce modèle, nous proposons d'étendre celui-ci par l'introduction d'une distribution aléatoire non uniforme d'impulsions. Considérant $\mathbf{p} = (X, 1)$, un point de dimension spécifiée situé aux coordonnées X, nous proposons la nouvelle formulation suivante :

$$n_s(\mathbf{p}) = \sum_{j} \delta\left(\xi(\mathbf{p}_j) < d(\mathbf{p}_j)\right) \, |w_j(\mathbf{p}_j)| \, K_j(\mathbf{p}) \tag{2.5}$$

Où $K_j(\mathbf{p}) = k_s\left((\mathbf{p} - \mathbf{p}_j)\, \mathbf{R}_s(\mathbf{p}_j)\, \mathbf{S}_s(\mathbf{p}_j)\right)$. d est un champ scalaire représentant une probabilité, δ correspond au Kronecker delta et $||$ à la valeur absolue. ξ est une variable aléatoire dont l'instance est indépendante de w_j. Le champ scalaire d permet de contrôler la densité d'impulsions sur des zones précises de l'espace de distribution. De par l'utilisation d'une valeur absolue, les régions à haute densité d'impulsions génèrent des valeurs de bruit proches de 1 tandis que les zones à faible densité d'impulsions génèrent des valeurs de bruit proches de 0.

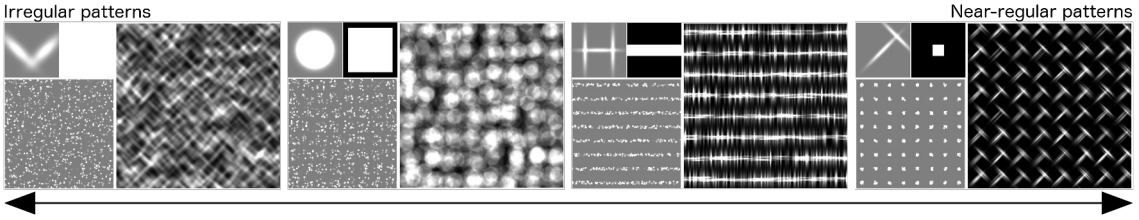


FIGURE 2.4 – Exemples de motifs produits par notre distribution.
Les motifs sont triés de gauche à droite du plus irrégulier jusqu'au quasi-régulier. Pour chaque exemple sur la colonne de gauche, l'image en haut à gauche est le noyau utilisé, l'image en haut à droite est le profil de densité périodique, et l'image du bas est la distribution d'impulsions obtenus. Les colonnes de droite de chaque exemple sont les motifs de bruit produits par ces configurations.

L'apparence globale de la texture produite est directement liée à l'aspect du d : l'énergie du motif est concentrée dans les zones de haute densité dans le champ scalaire de densité. Ce champ introduit un nouveau niveau de contrôle sur l'apparence de la texture générée, et peut être utilisé pour créer une structure globale à grande échelle. Pour créer une régularité structurelle, nous définissons d soit comme un champ de densité périodique tuilant l'espace d'évaluation (tel que dans les figures 2.4, 2.8 et 2.9), soit comme un champ de densité globale (tel que dans la figure 2.10). Ces deux types de champs peuvent également être combinés pour créer différents niveaux de structure dans l'image, tels que la structure complexe présentée dans la figure 2.5. Par simplicité, les champs de densité périodique présentés dans les différentes figures sont représentés par la densité d'impulsions sur une unique période, à laquelle nous faisons référence sous le nom de *profil de densité* ou *profil de distribution*. Ces profils sont créés en utilisant des fonctions de forme simple (telles qu'un rectangle paramétrique dans les différentes figures), ces fonctions permettant une édition interactive et une évaluation rapide de la densité d'impulsions. La figure 2.4 démontre les possibilités offertes par ce contrôle sur la distribution d'impulsions grâce à différents profils de densité périodique. Notre modèle est capable de produire un large panel de motifs, allant de motifs irréguliers (figure 2.4 à gauche) jusqu'à des motifs quasi réguliers (figure 2.4 à droite).

Dans le cas des motifs quasi réguliers, malgré la régularité structurelle, la texture générée conserve des variations aléatoires : les impulsions sont toujours générées par un générateur aléatoire, seule la densité d'impulsions change spatialement. D'après nos expérimentations, la modélisation du profil de densité permet un contrôle aisé et intuitif de la structure de la texture : des alignements structurels provoquent de petits vides irréguliers ou des variations alignées dans le motif. Ces indices visuels permettent d'estimer une marge de positionnement des éléments dans une zone et de recréer un profil de densité concordant. L'utilisateur peut également éditer le noyau en temps réel, lui donnant un retour visuel instantané pendant la création ou la modification du motif.

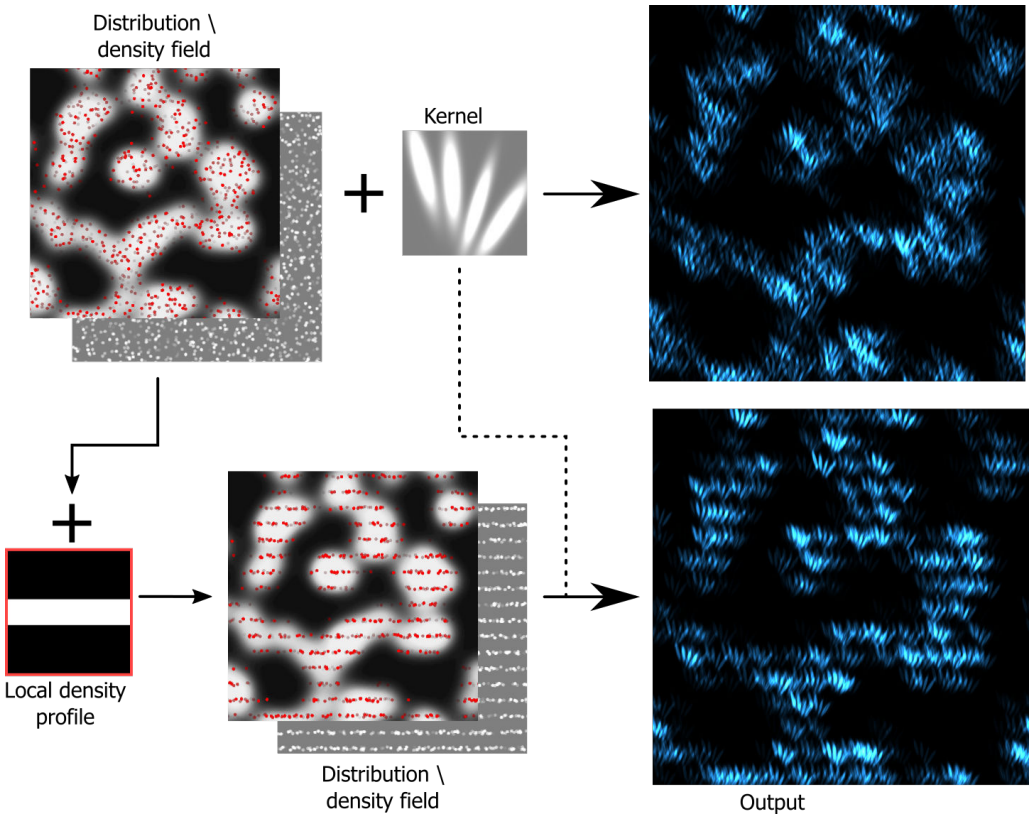


FIGURE 2.5 – Exemple d'utilisation combinée de profils de densité périodique et apériodique. Le champ scalaire de densité est utilisé pour rejeter les impulsions selon un test de densité et créer une structure globale dans l'image, produisant le motif en haut à droite par distribution aléatoire du noyau au centre de la première ligne. Le profil de densité périodique contrôle l'espace de distribution locale, ce profil restreignant dans la pratique l'espace de distribution pour concentrer les impulsions en des zones précises. Il en résulte un motif (en bas à droite) comprenant différents niveaux de structures.

2.3 Synthèse de texture par analyse d'exemple

Comme présenté dans la section précédente, notre modèle de bruit permet de créer certains types d'éléments structurels en particulier. Nous avons également montré que la structure peut être induite par le noyau d'évaluation ou par la distribution d'impulsions, en utilisant une distribution non uniforme de celles-ci. Cependant, analyser un motif donné pour en extraire à la fois la distribution et la forme d'un noyau est une tâche complexe : ces deux caractéristiques structurelles d'un motif sont intrinsèquement liées et peuvent difficilement être séparées. Une solution consiste à fixer l'une des deux pour en retrouver l'autre. Nous proposons dans notre cas une approche similaire à celle du bruit local à phase aléatoire : nous extrayons des noyaux de formes complexes à partir d'exemples, ces noyaux étant ensuite distribués sur une grille régulière. Nous rappelons d'abord brièvement le processus d'analyse utilisé par le bruit local à phase aléatoire pour son paramétrage automatique, prévu pour le traitement de certains types de textures structurées.

2.3.1 Paramétrage automatique du bruit local à phase aléatoire

Le processus d'analyse du bruit local à phase aléatoire se base sur une segmentation du spectre de fréquence pour extraire les phases et magnitudes des éléments structurels : un spectre donné est stratifié selon des niveaux d'énergie, puis subdivisé en sous-strates pour le calcul des bruits locaux.

Une strate R correspondant à la zone contenant la plus grande quantité d'énergie est considérée comme définissant la structure du motif. Cette région du spectre est plus généralement considérée comme l'ensemble des fréquences « contenant la structure » et est modifiable par un paramètre $r \in [0; 1]$, ce dernier définissant la proportion de l'énergie totale contenue dans R. Les éléments structurels les plus important sont préservés dans la texture produite en fixant les phases et amplitudes des fréquences correspondantes. Aux valeurs limites du paramètre r, la texture générée varie de complètement procédurale ($r = 0$) à la copie exacte de l'image donnée en entrée ($r = 1$). En pratique et pour les textures aléatoires faiblement structurées, Gilet *et al.* rapportent qu'une valeur de $r \approx 20\%$ est suffisante pour la préservation des éléments structurels et des variations aléatoires du motif. Le calcul final du bruit est réalisé par l'addition des bruits, approximant la totalité de l'énergie des strates.

$$n = n_R + \sum_S n_S \tag{2.6}$$

Deux types de bruits sont alors considérés : les bruits n_S dépendent entièrement du spectre de puissance et de phases aléatoires (nécessitant une quantité de mémoire minimale), tandis que le bruit n_R utilise des phases et amplitudes fixes.

Un inconvénient induit par la strate R est qu'un grand nombre de cosinus sont nécessaires pour représenter précisément le bruit n_R, ce qui se traduit par un surcoût d'évaluation important. Gilet *et al.* limitent ce surcoût par un compromis sur la continuité. L'image de la structure (la **DFT** inverse de n_R) est décomposée de manière itérative en une grille de blocs. Une **FFT** réalisée par bloc de cette image est calculée et un nombre fixé

de fréquences à hautes amplitudes est sélectionné et stocké pour chaque bloc. Le bruit n_R est finalement évalué par bloc dans le domaine spatial et réassemblé durant le rendu en utilisant une fonction de fenêtrage.

2.3.2 Reproduction de structures par bruit de spot localement contrôlé

Il est d'abord nécessaire de séparer la partie structurelle de l'image à reproduire de sa partie gaussienne (aléatoire). Pour cela, nous utilisons l'approche du bruit local à phase aléatoire, c'est-à-dire la stratification du spectre de puissance et l'extraction de l'image structurelle par **DFT** inverse de la région du spectre contenant la plus grande quantité d'énergie. Le but de notre analyse est de paramétrer un ensemble de I noyaux locaux, encodant chacun une partie de la structure. Nous calculons ce paramétrage par subdivisions successives de l'image structurelle selon une grille régulière de résolution arbitraire, et la transformation de chaque bloc en une représentation basée sur des gaussiennes.

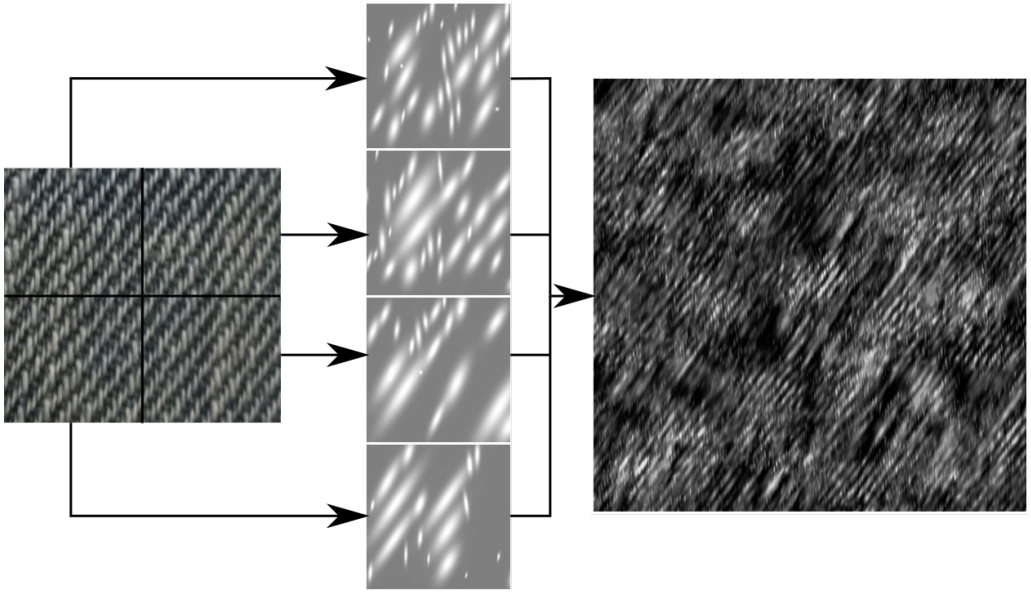


FIGURE 2.6 – Bruit de spot paramétré par analyse automatique d'un exemple.
Une image à reproduire (image de gauche) est subdivisée en échantillons plus petits. Chaque échantillon est ensuite décomposé en fonctions gaussiennes (≈ 20 pour cet exemple), ces fonctions formant un noyau spécifique pour chacun de ces échantillons (colonne centrale). Le motif de droite est obtenu par une distribution aléatoire des noyaux, et est généré en ≈ 100 millisecondes.

Cette approche revient à calculer une représentation compacte basée sur des gaussiennes elliptiques d'une image, ce qui est difficile pour des images complexes contenant par exemple des structures très irrégulières. En utilisant un algorithme d'*ellipse fitting* tel que celui proposé par [AWF95], les pixels de l'image sont approximés par J ellipses, celles-ci étant traduites en fonctions gaussiennes elliptiques de rayon correspondant. Ces fonctions sont la base du noyau utilisé par notre bruit de spot tel que présenté dans la figure 2.6. L'efficacité de cet algorithme est fortement dépendante de la complexité de l'image, et fonctionne mieux sur des éléments structurels simples tels que ceux reproduits dans les figures 2.6 et 2.7. Une analyse en profondeur et une segmentation de l'image pourraient améliorer la qualité de l'approximation produite et pourraient reproduire des éléments structurels plus complexes.

Le nombre J de fonctions gaussiennes est constant pour chaque bloc de l'image et impacte la précision de l'approximation et la vitesse de rendu du bruit de spot. Cette vitesse de rendu est linéairement dépendante du nombre de fonctions gaussiennes composant les noyaux. Le compromis entre vitesse et précision est un paramètre de notre modèle, l'utilisateur pouvant modifier J selon ses besoins. Dans les différentes figures de ce chapitre, ce nombre varie entre 4 et 8 pour les exemples simples, et monte jusqu'à 20 pour la création de noyaux complexes.

2.3.3 Combinaison de la structure et du bruit

Pour le rendu du bruit de spot, les impulsions sont distribuées en utilisant un *jittering* de grille régulière d'échantillons, c'est-à-dire par déplacement aléatoire des sommets d'une grille de positions entières (voir section 1.2.1). La résolution de la grille d'échantillons correspond à la résolution utilisée lors du processus de subdivision de l'image structurelle. Chaque impulsion est associée à un noyau, ce noyau pouvant être choisi comme le noyau approximant le bloc de l'image structurelle correspondant dans la grille régulière, ou pouvant être choisi aléatoirement pour augmenter les variations aléatoires. En utilisant le noyau approximant un bloc structurel, les éléments structurels correspondant aux basses fréquences peuvent être représentés par la combinaison de noyaux dans le résultat, au prix d'une diminution des variations. La partie gaussienne (aléatoire) de la texture à reproduire est ensuite ajoutée au résultat en utilisant un bruit basé sur un noyau construit à l'aide de cosinus, similaire au bruit local à phase aléatoire, ou par la distribution aléatoire de noyaux simples tels que ceux définis dans la figure 2.2.

2.4 Résultats

Nous avons implanté notre bruit sur carte graphique via un *fragment shader* en OpenGL. Le générateur de nombres aléatoires utilisé est un **PRNG** congruentiel linéaire initialisé par un code de Morton similaire à celui utilisé par Lagae *et al.* pour le bruit de Gabor [LLDD09]. Les performances sont fortement dépendantes de la densité d'impulsions et de la complexité des noyaux utilisés. Les différentes figures de ce chapitre sont générées entre 6 millisecondes et 100 millisecondes, dans une fenêtre de résolution 1200×1200 pixels sur une GeForce GTX 980. Les figures 2.6 et 2.7 présentent des exemples de reproduction de structures depuis une image donnée. La forme simple des éléments structurels est reproduite de manière précise par notre processus d'analyse. Comme précisé précédemment, notre méthode repose sur une segmentation automatique et l'évaluation d'une représentation basée sur des fonctions gaussiennes d'une image. Une méthode d'*ellipse fitting* directe produit de bons résultats pour des motifs simples, mais l'analyse automatique d'un motif complexe reste un problème difficile. Nous pensons cependant que cette approche est un premier pas vers la génération de textures procédurales par des bruits de spot contrôlés via une analyse entièrement automatique d'un motif.

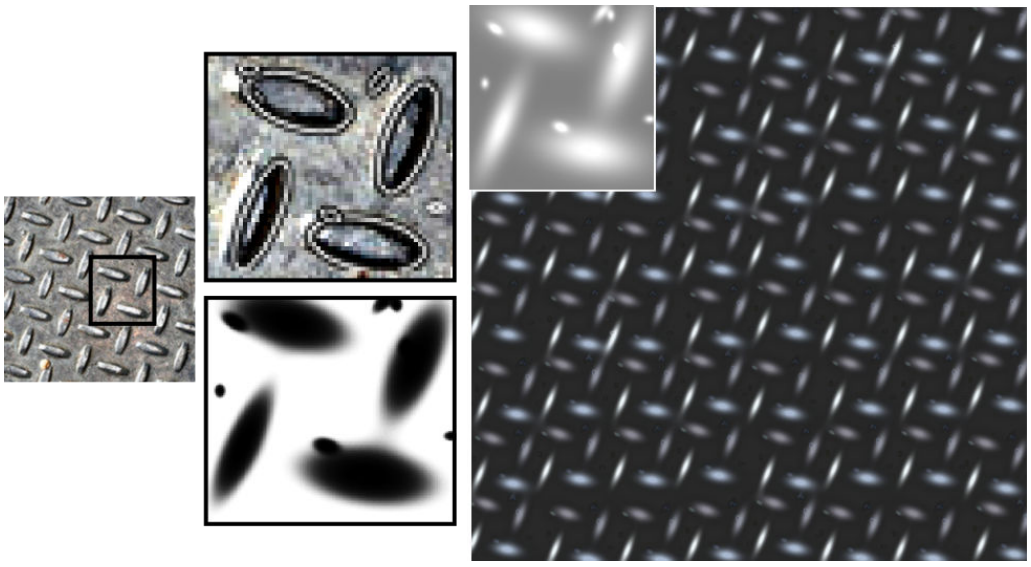


FIGURE 2.7 – Exemple de motif simple produit grâce à l'analyse d'un échantillon d'une image. L'échantillon du motif est analysé par un algorithme d'*ellipse fitting*, produisant une somme de fonctions gaussiennes (au centre) utilisée pour générer la structure de droite. Le résultat est généré en ≈ 6 millisecondes.

Plusieurs motifs peuvent également être créés par la définition manuelle d'un noyau. La figure 2.8 présente différents exemples de motifs reproduits grâce à la définition d'une distribution périodique et d'un noyau adéquat édité par l'utilisateur.

Contrairement à la majorité des bruits présentés dans la section 1.3, notre modèle se focalise sur l'édition spatiale de motifs : la modélisation visuelle d'un motif est plus aisée que l'édition directe du spectre de puissance. La figure 2.9 présente un exemple de chaîne d'édition, et montre comment la modification interactive du noyau et de la distribution impacte le résultat global, permettant à un utilisateur d'obtenir un retour visuel direct de

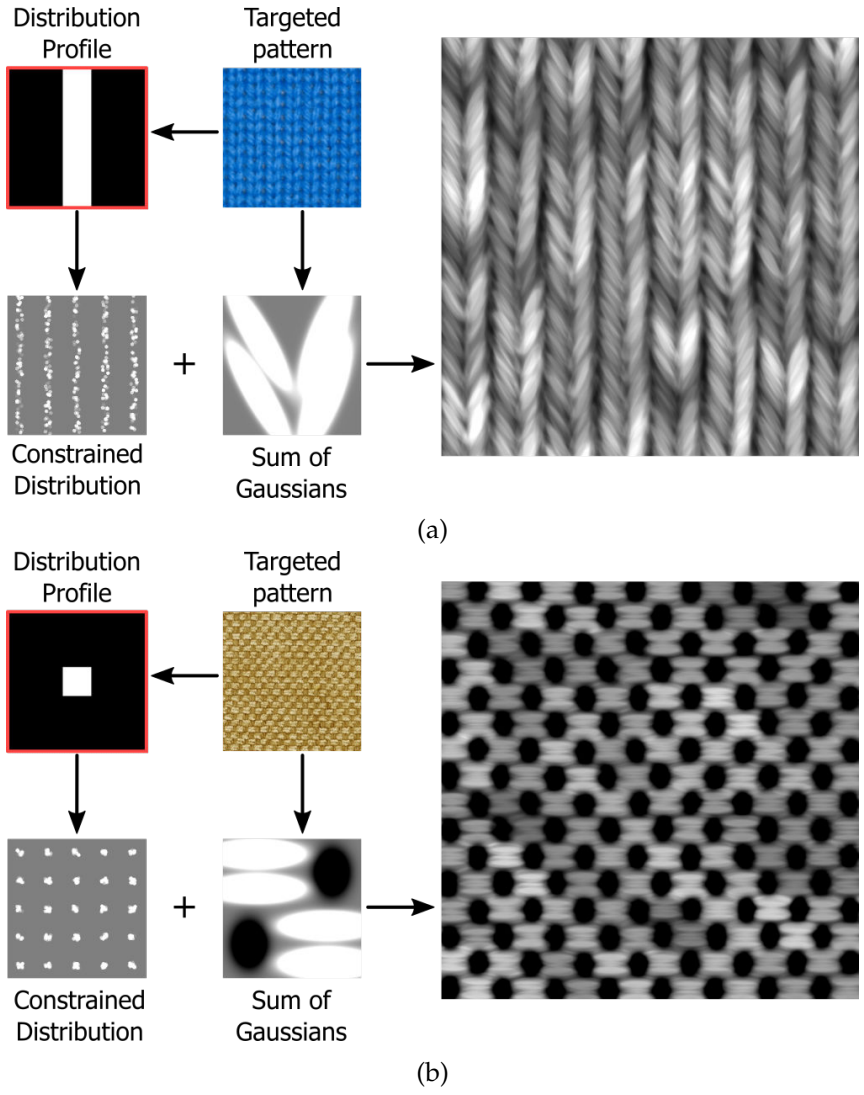


FIGURE 2.8 – Exemple de reproduction d'éléments structurels quasi réguliers par un unique bruit de spot. Pour ces deux exemples, le profil du noyau représenté par a somme de gaussiennes (milieu de la deuxième ligne) est fourni pas une décomposition d'un échantillon de la texture cible (milieu de la première ligne) guidée par l'utilisateur. Le profil de distribution (en haut à gauche) est défini directement dans le modèle, mais peut également être défini par une texture. Ce profil génère une distribution d'impulsions présentée en bas à gauche dans chaque figure. L'image (a) est générée en $\approx$ 17 millisecondes et l'image (b) en $\approx$ 25 millisecondes.

ses modifications. La figure 2.10 présente les possibilités de contrôle de notre modèle de bruit sur différentes distributions de structures dans un unique motif. Les champs scalaires de rotation $\mathbf{R}_s$ et de changement d'échelle $\mathbf{S}_s$ de l'équation 2.2 permettent de créer des motifs incluant des variations locales complexes, telles que celles présentées dans la figure 2.11 utilisant un champ scalaire pour influencer une rotation aléatoire définie par $\mathbf{R}_s$. La figure 2.12 présente une application d'un motif sur des objets 3D en utilisant un simple *bump mapping* pour créer un relief. La figure 2.12.a utilise la configuration du motif (a) de la figure 2.8 pour le calcul du bruit, lui-même utilisé comme un champ de hauteurs. Les normales utilisées pour le *bump mapping* sont calculées par différences finies entre trois évaluations du bruit dans le *fragment shader*.

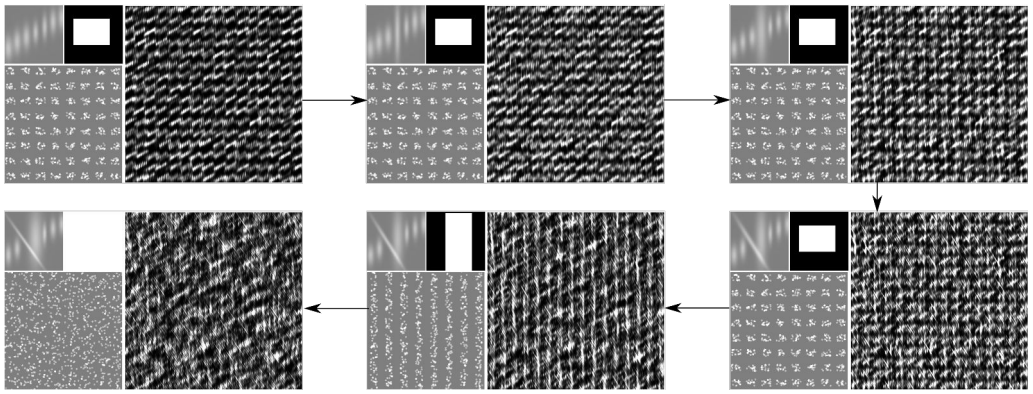


FIGURE 2.9 – Exemple de processus d'édition du bruit de spot localement contrôlé.
A partir d'un noyau et d'une distribution donnée (en haut à gauche), l'utilisateur peut modifier interactivement chaque fonction gaussienne du noyau et le profile de distribution pour éditer l'apparence du résultat. Les motifs créés par cette configuration sont rendu en ≈ 22 millisecondes.

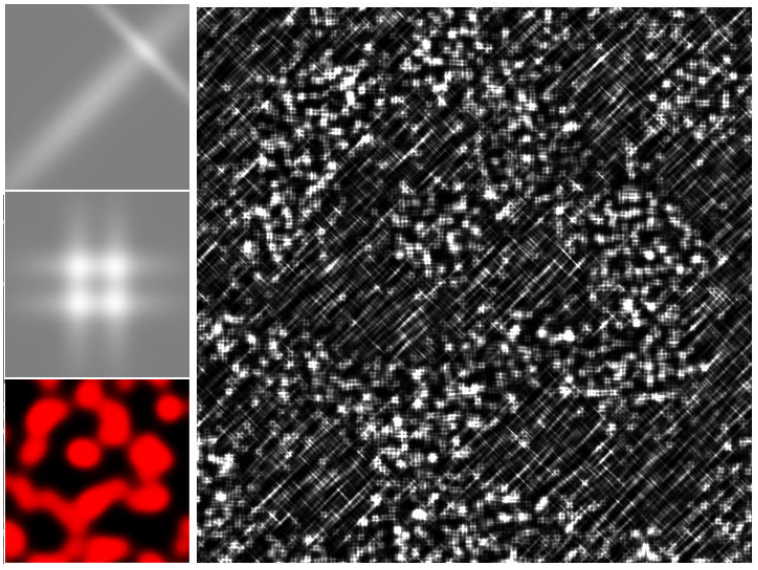


FIGURE 2.10 – Exemple de noyaux mélangés pour la répartition de structure dans un unique motif de bruit.
Un champs de densité globale (en bas à gauche) est utilisé pour contrôler la distribution des noyaux : pour une impulsion généré, le noyau correspondant (profiles en haut et au milieu à gauche) est sélectionné selon un test de densité aléatoire. Le résultat est généré en ≈ 18 millisecondes.

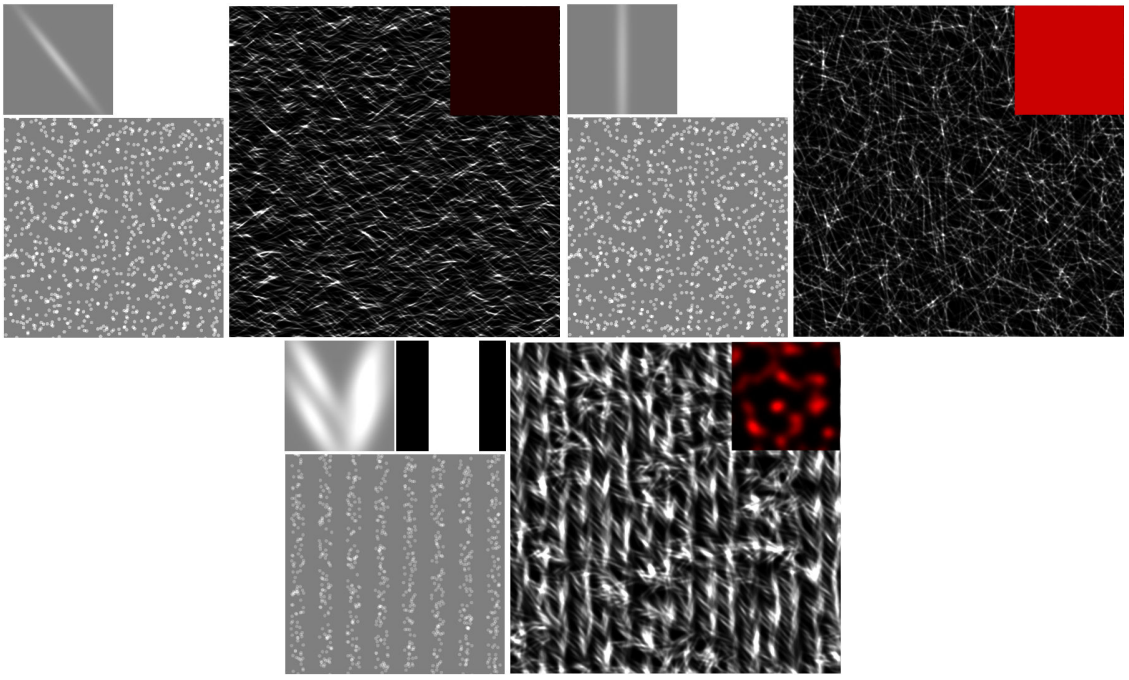
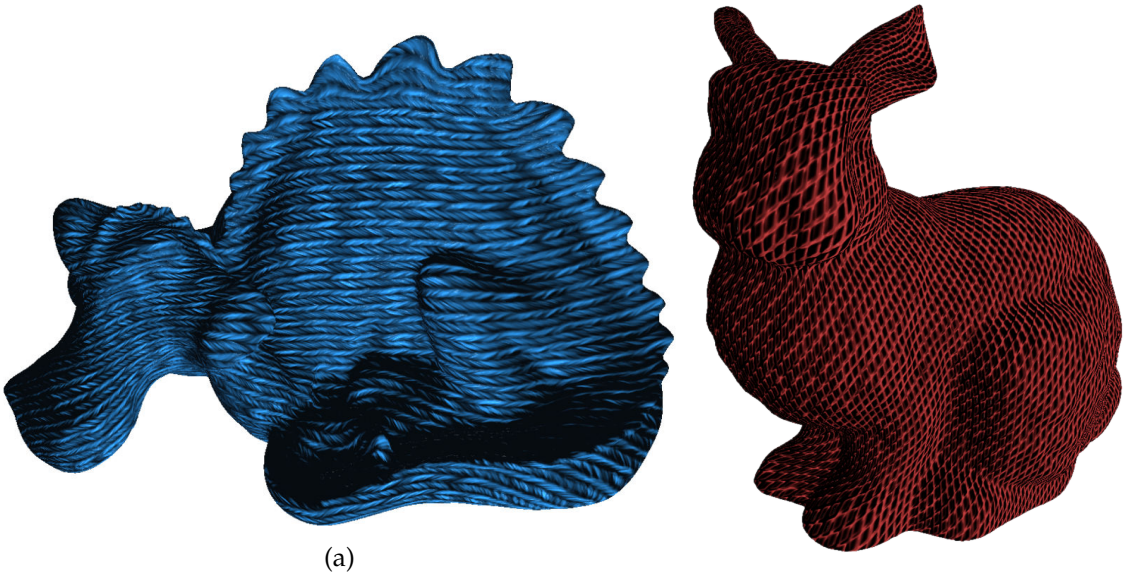


FIGURE 2.11 – Exemple d'utilisation d'un champ scalaire pour la définition de degrés de rotation des impulsions.
Un champ scalaire (en rouge en haut à droite) est utilisé pour contrôler la rotation $\mathbf{R}_s$, définit dans ces exemple par une fonction de rotation aléatoire entre 0 et 2π multipliée par la valeur du champ d'intensité à la position de l'impulsion.



(a) (b)

FIGURE 2.12 – Exemples de motifs procéduraux évalués sur des objets.
La fonction de bruit est utilisée pour calculer les hauteurs, elles-mêmes utilisées par le *bump mapping*. La figure (a) est basée la configuration du motif de tissu bleu présentée dans la figure 2.8.a.
Ces deux résultats sont générés en ≈ 50 millisecondes.

2.5 Conclusion

Nous avons présenté dans ce chapitre un nouveau modèle de bruit basé sur un bruit de spot localement contrôlé permettant de reproduire des motifs irréguliers et quasi réguliers. Les éléments structurels quasi réguliers sont produits par combinaison d'un noyau structuré et d'une distribution aléatoire contrôlée. Ce modèle de bruit, étendant le bruit local à phase aléatoire, peut toujours produire des motifs irréguliers comportant des éléments de structure.

Notre fonction de bruit diffère de celles présentées dans la section 1.3, nos travaux se focalisant sur le contrôle spatial plutôt que sur la reconstruction d'un spectre de puissance : éditer interactivement la forme d'un motif à travers le domaine spatial est plus facile pour un artiste, comparativement à l'édition d'un spectre de puissance donnée. Les bruits sont difficiles à contrôler, et ne sont généralement pas adaptés à la modélisation de textures procédurales structurées. Notre bruit de spot localement contrôlé peut cependant apporter une première réponse aux difficultés de modélisation de ce type de texture procédurale.

Comme précisé dans la section 2.3.2, l'algorithme utilisé pour l'analyse d'un exemple peut efficacement extraire les paramètres pour la reproduction d'une structure simple, mais la reproduction d'une structure complexe reste difficile et peut nécessiter un nombre important de gaussiennes. Une méthode d'analyse plus poussée, ou une segmentation de l'exemple, pourrait permettre d'améliorer cette décomposition. D'autres approches de la décomposition pourraient également être utilisées pour remplacer ou assister l'algorithme d'*ellipse fitting*. Par exemple, une décomposition en ondelettes de Gabor telle que celle proposée par Welsh et Mueller [WM03] pourrait être utilisée pour décomposer l'exemple en une arborescence de fonctions gaussiennes représentant différents niveaux de compression de la représentation des éléments. Cette dernière approche permettrait en sus de produire un noyau multi-échelles pour une visualisation interactive optimisée.

Une autre extension possible pour limiter le nombre de fonctions gaussiennes serait d'utiliser des courbes paramétriques, telles que des courbes de Bézier ou de Hermite (section 1.1.1), et de les associées à un profil gaussien pour créer une nouvelle primitive de construction du noyau. Ces courbes permettraient à un utilisateur de créer des détails de surface continus de formes très variées, et pourraient grandement améliorer la qualité de modélisation des éléments tels que les craquelures. La décomposition en ellipse pourrait être adaptée à cette extension de la formulation du noyau en utilisant, par exemple, des algorithmes de squelettisation de l'image pour extraire les formes des courbes automatiquement.

Chapitre 3

Le Bruit de Spot Procédural de Surcouche pour la Génération de Détails Stochastiques de Surface

Sommaire

Chapitre 3

Le Bruit de Spot Procédural de Surcouche pour la Génération de Détails Stochastiques de Surface

Ces travaux ont fait l'objet d'une soumission à la conférence internationale *Computer Graphics and Visual Computing* 2016 [PGD*16].

3.1 Introduction

Les artistes tentent de se rapprocher du photo-réalisme en augmentant autant que possible la complexité visuelle et la quantité de détails des objets des scènes virtuelles. Une grande quantité de détails géométriques de petites tailles sur les surfaces doit être définie puis rendue pour obtenir ce résultat. L'édition et le rendu temps réel de scènes vastes, comportant plusieurs millions de détails de petites et moyennes tailles, restent des tâches difficiles à accomplir, en particulier pour les détails tels que l'herbe ou la fourrure. Une couleur de surface ou un *bump mapping* ne permettent pas un rendu réaliste de ces objets, d'importants effets de parallaxe manquant à l'image produite et de nombreux artefacts devenant visibles de près. L'instanciation massive de primitives géométriques est également inadaptée pour le rendu de ce type de détails, à cause des contraintes de stockage, du temps de rendu prohibitif et des difficultés de filtrage et de gestion de la transparence.

Le plaquage de surcouches (l'application de textures semi-transparentes sur différentes couches empilées au-dessus de la surface, voir section 1.1.3) évite la plupart de ces limitations. La texture volumique représentant ces couches peut être stockée sous la forme d'un tableau de données explicites (voxels) ou par des fonctions analytiques continues. Les fonctions analytiques telles que les bruits procéduraux (voir section 1.3) apportent d'importants avantages, les rendant très attractives pour la modélisation de détails 3D formant une apparence chaotique dans l'image : ces fonctions génèrent des détails de manière compacte et continue, permettant une visualisation proche de la ca-

méra de ces derniers sans artefacts de voxelisation. Leurs formulations restent génériques et permettent la définition de variations locales par modification des paramètres du bruit.

L'utilisation des bruits procéduraux pour la création de textures procédurales reste limitée dans les applications graphiques : la modélisation d'un motif en utilisant un bruit procédural est une tâche complexe, et le rendu de textures de surcouche requiert des méthodes de rendu volumétrique direct coûteuses en performances. Le modèle de bruit introduit dans le chapitre 2, basé sur une distribution non uniforme d'impulsions associées à des noyaux de convolution définis spatialement, facilite le processus d'édition artistique. Mais ce modèle est dépendant de la paramétrisation de l'espace texture, dans lequel sont distribuées les impulsions.

Nous présentons dans ce chapitre une extension du bruit de spot localement contrôlé pour la modélisation de détails de surface stochastiques 3D, composés de nombreux éléments quasi similaires. Nous utilisons une convolution éparse 2D projective de noyaux 3D filtrés, définis par une somme de gaussiennes anisotropes orientées arbitrairement, pour modéliser une texture de surcouche. Pour la visualisation interactive du motif produit par ce bruit, nous proposons une méthode d'évaluation optimisée de la contribution des noyaux à un rayon, ainsi que deux méthodes de rendu volumétrique :

- Une première méthode basée sur un échantillonnage pas à pas d'une grille par un algorithme de Bresenham 3D, pour un rendu interactif de détails de surface extrêmement denses (plusieurs dizaines de millions d'éléments).
- Une seconde méthode basée sur la projection et la rastérisation de noyaux instanciés, pour une évaluation plus rapide et une édition interactive de motifs lorsque la densité de détails à l'écran est moindre.

Nous exploitons le fait que notre modèle de bruit de spot utilise une convolution éparse de noyaux dont le support est fini. La formulation du noyau, basée sur des gaussiennes, nous permet une évaluation accélérée en rendu projectif similaire au rendu par projection de voxels (voir section 1.4.2).

Cette extension du modèle de bruit associée à ces méthodes de rendu nous permet une meilleure interactivité pour l'édition de textures de surcouche procédurales. Nous illustrons les possibilités offertes par ces contributions par leur utilisation pour la modélisation et le rendu de champs d'herbe et de fourrures (figures 3.8, 3.7 et 3.13), mais également de noyaux volumiques plus complexes (figure 3.15).

3.2 Le bruit procédural volumique de surcouche

Pour rappel, notre précédent modèle de bruit se base sur l'utilisation d'un noyau défini par une somme de gaussiennes elliptiques :

$$k(\mathbf{p}) = \sum_{V_i} g_{V_i} \qquad g_V(\mathbf{p}) = Ae^{-\frac{1}{2}\mathbf{p}^T\mathbf{V}^{-1}\mathbf{p}} \tag{3.1}$$

$\mathbf{p}$ est un point en dimension $N+1$ dont la dernière coordonnée est égale à 1, N désignant la dimension de l'espace d'évaluation. $\mathbf{V}$ est une matrice carrée semi-positive de dimension $(N+1)\times(N+1)$ décrivant l'isocontour de l'ellipse et A est l'amplitude de l'enveloppe gaussienne. La distribution non uniforme est obtenue par l'utilisation d'une distribution aléatoire contrainte par l'utilisation d'un delta de Kroenecker (δ), ce dernier rejetant les impulsions dont le facteur de densité aléatoire est inférieur à un facteur de densité local. Le modèle final de bruit est donné par l'équation :

$$n_s(\mathbf{p}) = \sum_j \delta\left(\xi(\mathbf{p}_j) < d(\mathbf{p}_j)\right) \left|w_j(\mathbf{p}_j)\right| K_j(\mathbf{p}) \tag{3.2}$$

Avec $K_j(\mathbf{p}) = k_s\left((\mathbf{p}-\mathbf{p}_j)\,\mathbf{R}_s(\mathbf{p}_j)\,\mathbf{S}_s(\mathbf{p}_j)\right)$, $\mathbf{R}_s$ et $\mathbf{S}_s$ correspondant respectivement à des matrices de rotation et de changement d'échelle contrôlées par des champs scalaires sous-jacents. $||$ correspond à la valeur absolue, d est un champ scalaire de probabilité contrôlant la densité locale d'impulsions et ξ est une variable aléatoire dont le tirage est indépendant de w_j.

Deux points sont négligés par ce modèle. Premièrement, le filtrage du bruit n'est pas détaillé dans la formulation précédente. L'utilisation d'une somme de gaussiennes permet cependant un filtrage analytique, en particulier si une méthode de rendu par projection de points est utilisée pour l'évaluation des différentes gaussiennes [ZPvBG01]. La distribution non uniforme proposée dépend de la paramétrisation de l'espace de distribution, à savoir l'espace texture dans le modèle précédent.

3.2.1 Le noyau multi-gaussiennes 3D filtrées

Nous étendons la précédente formulation générique du noyau pour la modélisation de textures de surcouche volumiques en définissant un noyau d'évaluation 3D filtré, basé sur une somme de gaussiennes ellipsoïdales arbitrairement orientées. La forme des noyaux peut être contrôlée par modification des paramètres des différentes fonctions gaussiennes ellipsoïdales le définissant. Nous considérons la fonction gaussienne filtrée utilisé par Zwicker *et al.* [ZPvBG01] comme nouvelle primitive de construction des noyaux :

$$g_V(\mathbf{p}) = \frac{||\mathbf{V}^{-1'}||^{1/2}}{(2\pi)^{3/2}} e^{-\frac{1}{2}\mathbf{p}^T\mathbf{V}^{-1}\mathbf{p}} \tag{3.3}$$

La matrice $\mathbf{V}$ est une matrice de dimension 4×4 construite selon le produit matriciel :

$$\mathbf{V}^{-1} = (\mathbf{M}\,\mathbf{R}\,\mathbf{S})^{-T}(\mathbf{M}\,\mathbf{R}\,\mathbf{S})^{-1} \tag{3.4}$$

$\mathbf{V}^{-1'}$ est la matrice $\mathbf{V}^{-1}$ réduite à une dimension 3×3, en ignorant les dernières lignes et colonnes de $\mathbf{V}^{-1}$ encodant la translation. $||\mathbf{V}^{-1'}||$ est le déterminant de cette matrice réduite. **M**, **R** et **S** sont respectivement des matrices de translation, de rotation et de changement d'échelle. L'isocontour de la gaussienne ellipsoïdale est donné par le produit $\mathbf{p}^T\mathbf{V}^{-1}\mathbf{p}$, décrivant une surface implicite quadratique si $\mathbf{V}$ est une matrice quadratique semi-positive ($\mathbf{p}$ est un point exprimé en coordonnées homogènes). L'équation 3.3 nous permet d'obtenir un filtre de reconstruction en incorporant la jacobienne du pixel projetée dans l'espace du noyau pour chaque fonction gaussienne, similairement à l'approche de Ren *et al.* [RPZ02]. La formulation de notre nouveau noyau devient alors :

$$k(\mathbf{p}) = \sum_{V_i} \rho'_{k,V_i} \qquad \rho'_{k,V}(\mathbf{p}) = g_{V+J_k^{-1}J_k^{-1^T}}(\mathbf{p}) \tag{3.5}$$

J_k est ici la jacobienne projetée dans l'espace du noyau, et $\rho'_{V,k}$ est le filtre de reconstruction d'une gaussienne g_V dans l'espace du noyau.

Cette formulation 3D du noyau nous permet de modéliser aisément des surfaces très fines telles que des brins d'herbe, en utilisant un unique noyau gaussien plat. Les structures semi-transparentes complexes, telles que la fourrure ou le coton, peuvent être modélisées par de multiples gaussiennes, similairement aux différentes textures de surcouche présentées dans la section 3.4.

3.2.2 Génération d'impulsions surfaciques à la volée

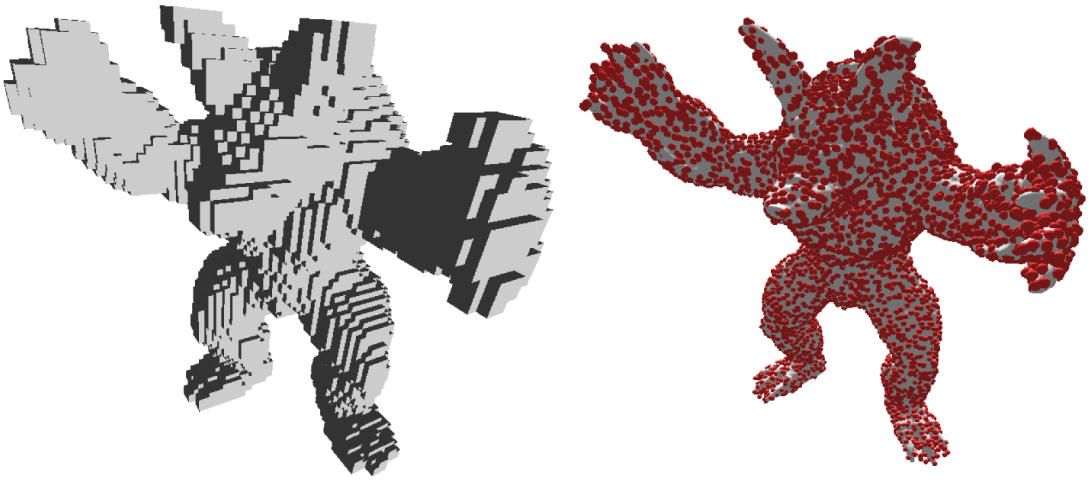


FIGURE 3.1 – Génération d'impulsions surfaciques.
Un *octree* (à gauche) est utilisé pour générer une distribution d'impulsion sur des surfaces arbitraires (à droite), les feuilles de l'*octree* correspondant aux voxels.

Pour réaliser la génération d'impulsions à la volée sur une surface arbitraire, celle-ci est partitionnée en une grille 3D régulière. La taille de chaque cellule de la grille est fixée selon la taille des noyaux, afin d'éviter qu'un noyau ne recouvre plus de deux cellules (un noyau situé au centre d'une cellule ne peut contribuer qu'aux cellules directement voisines). Similairement à Lefebvre *et al.* [LHN05] et Lagae *et al.* [LLDD09], nos cellules

sont des voxels couvrant la surface. Les noyaux sont distribués dans un espace 3D en utilisant un **PRNG** initialisé par un indice unique correspondant au voxel. Les impulsions 3D générées sont ensuite projetées sur un plan approximant la surface couverte (figure 3.1). Pour minimiser l'espace mémoire nécessaire, une structure d'*octree* est utilisée. Les nœuds terminaux de l'*octree* sont les voxels pour lesquels sont stockés un vecteur $\mathbf{N}(u, v)$, pointant vers le point de la surface le plus proche $P(u, v)$, et une tangente à la surface en ce point $T(u, v)$. $-\mathbf{N}/||\mathbf{N}||$ est la normale au point P. Le noyau K est ensuite orienté selon ces propriétés de surface (figure 3.2). La position d'une impulsion peut également être éloignée de la surface par décalage le long de la normale, cette position étant notée $P(u, v, h)$ dans la figure 3.2. L'utilisation d'un *octree* permet également d'utiliser des *octree textures* pour la définition de paramètres locaux utilisés par les noyaux, tels que la couleur, l'orientation et la taille de ceux-ci.

Le nombre d'impulsions dans une cellule est défini aléatoirement selon la densité locale d'impulsion définie pour cette cellule (chaque cellule contient une information de densité d). Similairement à Lagae *et al.* [LLDD09], ce type de bruit indépendant de la paramétrisation de la surface part de l'hypothèse que les détails de texture sont petits comparativement aux détails géométriques, ce qui est une hypothèse courante lors de l'utilisation du plaquage de texture.

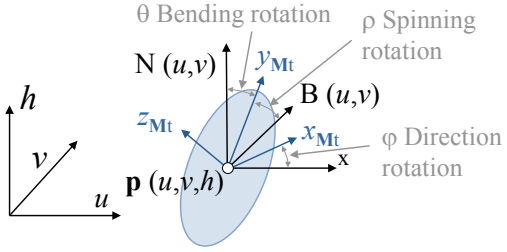


FIGURE 3.2 – Espace d'évaluation du noyau transposé dans l'espace de surcouche (u, v, h).

3.3 Rendu de bruit de spot volumique de surcouche

Le bruit de spot volumique de surcouche précédent définit des champs de couleur et de transparence volumiques (n_{rgb} et n_a). Une méthode de rendu volumétrique est nécessaire pour évaluer le transport de la lumière dans cette surcouche procédurale.

3.3.1 Transport de la lumière dans la surcouche

L'équation du rendu volumétrique décrit l'intensité de la lumière L atteignant un point de projection (par exemple un pixel) le long du rayon l. Pour le rendu interactif, une approximation commune consiste à négliger la diffusion dans ce volume, considérant uniquement l'émission et l'absorption de la luminance par la matière du volume. Une possible formulation de ce phénomène est :

$$L = \int_0^l (\mathbf{E} \circ n)(\mathbf{p}(s)) e^{-\mathbf{A}_0(s)} \delta s \qquad \mathbf{A}_0(s) = \int_0^s n_a(\mathbf{p}(t)) \delta t \tag{3.6}$$

Où l'exponentielle représente l'épaisseur optique (voir section 1.4.1) et $(\mathbf{E} \circ n)(\mathbf{p}(s))$ est le terme décrivant l'émission et l'atténuation de la luminance dans la direction du rayon au point $\mathbf{p}(s)$. Si aucune illumination n'est considérée, ce terme correspond dans notre cas à :

$$(\mathbf{E} \circ n)(\mathbf{p}(s)) = n_{rgb}(\mathbf{p}(s))\, n_a(\mathbf{p}(s)) \tag{3.7}$$

Où $n_{rgb}(\mathbf{p}(s))$ est la couleur et $n_a(\mathbf{p}(s))$ est le coefficient de transmission renvoyés par la fonction de bruit au point $\mathbf{p}(s)$. Nous utilisons ci-après la fonction $\mathbf{E}$ pour exprimer la multiplication de la composante de couleur par la composante d'atténuation. Pour l'ombrage, le terme d'émission n_{rgb} doit être modifié par multiplication d'un facteur supplémentaire, généralement obtenu par produit scalaire entre le gradient de $n_a(\mathbf{p}(s))$ et la direction de la lumière [GAMD10]. Pour simplifier notre propos, nous présentons notre formulation sans calcul de l'ombrage.

L'évaluation du bruit de surcouche est optimisée par la définition d'une grille volumique régulière afin d'accélérer la recherche des noyaux les plus proches. Considérant un rayon de vue comme traversant individuellement les cellules, l'équation précédente peut être réécrite en décomposant l'espace évalué en une succession de cellules disjointes $\mathscr{C}_i$. En intégrant dans celle-ci le terme d'atténuation de l'opacité $(1 - \alpha_j)$ couramment utilisé dans le rendu volumétrique, l'équation 3.6 devient :

$$L \approx \sum_{i=0}^{N-1} L_{\mathscr{C}_i} \prod_{j=0}^{i-1} (1 - \alpha_j) \tag{3.8}$$

Où N est le nombre de cellules traversées par le rayon. $L_{\mathscr{C}_i}$ est la lumière diffusée par la cellule $\mathscr{C}_i$ dans la direction du rayon. Cette diffusion est exprimée par l'équation :

$$L_{\mathscr{C}_i} = \int_{l_i}^{l_{i+1}} (\mathbf{E} \circ n)(\mathbf{p}(s)) e^{-\mathbf{A}_{l_i}(s)} \delta s \tag{3.9}$$

Dans les parties suivantes, nous ne considérons que le cas d'une distribution d'impulsions uniforme, c'est-à-dire pour $\delta = 1$. En intégrant l'équation 3.2 dans l'équation 3.9 et en utilisant une approximation standard d'ordre 0 [HMDM08] du facteur d'atténuation, la contribution d'une cellule $\mathscr{C}_i$ peut être reformulée en :

$$L_{\mathscr{C}_i} \approx \sum_{u=0}^{U} \left(\mathbf{E} \circ \sum_{\mathbf{p}_j \in \mathscr{C}_i^*} w_j K(\mathbf{p}(u) - \mathbf{p}_j) \right) (\mathbf{p}(u)) \prod_{v=0}^{u-1} (1 - \alpha_v) \tag{3.10}$$

Où U est le nombre d'échantillons évalués le long du rayon traversant la cellule et $\mathbf{p}_j \in \mathscr{C}_i^*$ l'ensemble des impulsions contribuant à la cellule $\mathscr{C}_i$, c'est-à-dire les impulsions de la cellule et de ses voisines directes.

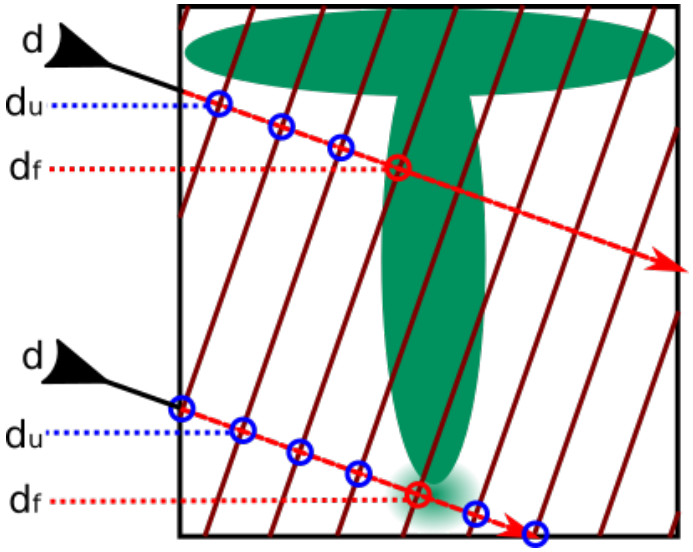
3.3.2 Évaluation optimisée des noyaux

Évaluer $L_{\mathscr{C}_i}$ en utilisant un algorithme d'échantillonnage pas à pas standard peut poser différents problèmes : la double somme de l'évaluation provoque un surcoût de calcul important, chaque noyau (lui-même constitué de plusieurs fonctions gaussiennes) étant évalué plusieurs fois. De plus, en raison de la limite de Nyquist, dans le cas extrême où les gaussiennes sont plates, tel que dans le cas des brins d'herbe, le pas entre les échantillons successifs doit être infiniment petit pour obtenir un rendu correct.

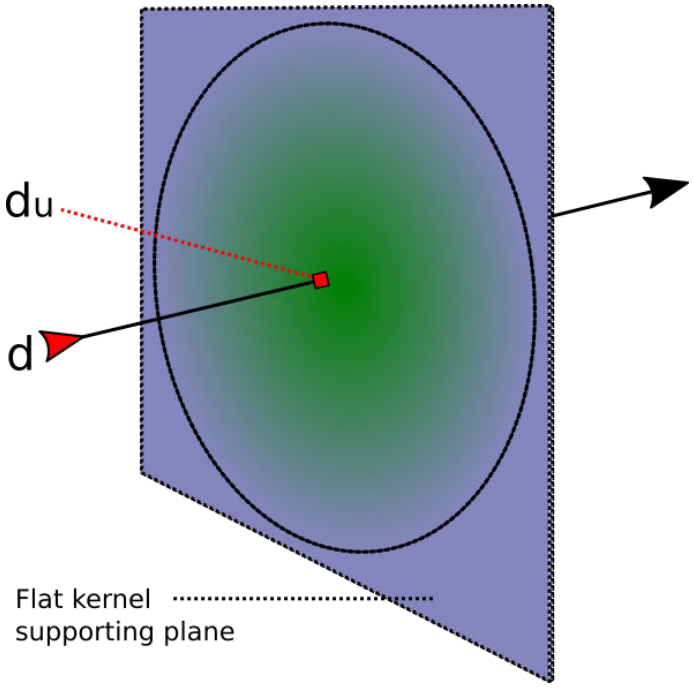
Pour optimiser les performances de rendu et éviter le problème d'échantillonnage dans le cas des gaussiennes plates, l'échantillonnage pas à pas de la cellule peut être remplacé par un échantillonnage optimisé des différents noyaux. Pour chaque noyau, nous considérons un échantillonnage pas à pas classique du rayon pour l'évaluation de la contribution, c'est-à-dire une somme d'échantillons discrets d_i situés le long du rayon :

$$\tilde{K}(\mathbf{d}, \mathbf{p}_j) \approx \sum_{u=0}^{N_s} (\mathbf{E} \circ w_j K(\mathbf{d}_u - \mathbf{p}_j)) \prod_{v=0}^{u-1} (1 - \alpha_v) \tag{3.11}$$

$K(\mathbf{d}_u - \mathbf{p}_j)$ est l'évaluation du noyau K pour l'impulsion $\mathbf{p}_j$ à la position d'un échantillon $\mathbf{d}_u$ dans le volume englobant de K. Le nombre N_s d'échantillons par noyau est directement lié à l'épaisseur de K. En pratique, le point de départ de l'échantillonnage est déterminé par intersection avec la boîte englobante. L'opacité et la couleur évaluées pour chaque position $\mathbf{d}_u$ sont accumulées jusqu'à ce qu'une opacité maximale soit atteinte ou que chaque échantillon ait été évalué (figure 3.3.a). Pour permettre une meilleure approximation de la profondeur de la contribution du noyau au rayon, la position de l'échantillon d_f à partir duquel s'arrête le rayon ou, dans le cas où une opacité maximale n'est pas atteinte, la position de l'échantillon le plus contributif au noyau est conservée. Dans le cas d'un noyau gaussien plat, un unique échantillon correspondant à l'intersection du rayon et du noyau est utilisé au lieu de la somme (figure 3.3.b).



(a) Noyau volumique



(b) Noyau plat

FIGURE 3.3 – Évaluation de la contribution d'un noyau au rayon.
Pour l'évaluation des noyaux volumiques (a), un échantillonnage pas à pas du rayon équivalant à une découpe en tranches successives est utilisé pour accumuler les contributions des échantillons situés aux positions d_u, jusqu'à l'atteinte d'une opacité maximale. En plus de cette contribution, la position de l'échantillon le plus contributif d_f (ou à partir duquel l'opacité maximale est atteinte) est conservée pour améliorer l'atténuation avec les autres noyaux. Dans le cas des noyaux plats (b), l'échantillonnage pas à pas du rayon est remplacé par l'évaluation de l'intersection du rayon avec le plan du noyau.

3.3.3 Rendu par échantillonnage de la grille

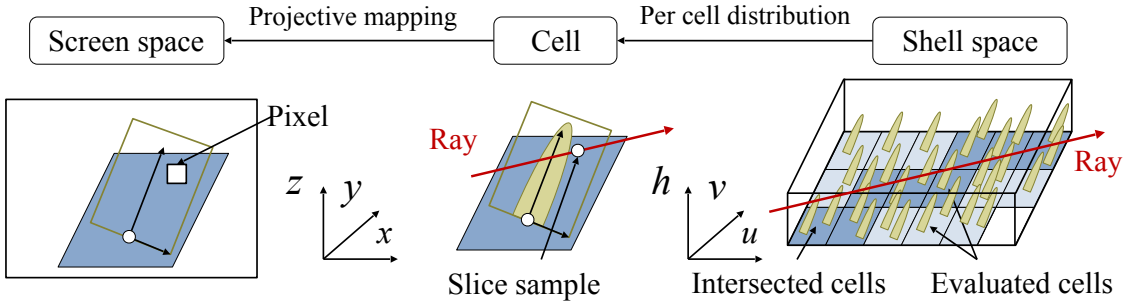


FIGURE 3.4 – Récapitulatif du rendu par échantillonnage pas à pas de la surcouche volumique.

Comme expliqué précédemment, le bruit de surcouche est optimisé par le découpage de l'espace d'évaluation du bruit en une grille régulière 3D. L'évaluation de L suivant l'équation 3.8 revient donc à évaluer chaque cellule traversée par le rayon (en bleu sur la figure 3.4). Ce parcours de la grille de cellule est réalisé dans notre cas de manière incrémentale par un algorithme de Bresenham. Pour chaque cellule parcourue, la distribution des noyaux est réalisée dans la cellule courante et dans ses voisines directes. Les voxels d'entrée et de sortie du rayon dans la surcouche peuvent être obtenus par rastérisation d'une extrusion de la surface. Pour limiter la redondance de calcul, un tableau est utilisé pour conserver les neuf derniers indices de cellules évaluées.

L'utilisation d'un échantillonnage pas à pas de la cellule tel que celui utilisé dans l'équation 3.10 multiplie les évaluations des différents noyaux distribués. L'équation 3.11 permet d'éviter cette surévaluation par une approche d'évaluation par noyau plutôt que par point : l'évaluation de l'opacité d'un noyau est réalisée individuellement par noyau selon une direction de vue $\mathbf{d}$. L'évaluation exacte de la contribution de la cellule nécessite alors un tri coûteux des noyaux selon leur profondeur dans l'écran. Dans le cas où l'atténuation exacte par cellule n'est pas nécessaire, tel que pour les objets semi-transparents quasi similaires, il est possible d'approximer la contribution de la cellule $L_{\mathscr{C}_i}$ par l'évaluation de la contribution moyenne des noyaux à la cellule courante. Cette contribution est obtenue par accumulation puis normalisation des contributions renvoyées par l'équation 3.11 :

$$L_{\mathscr{C}_i} = \frac{\sum_{\mathbf{p}_j \in \mathscr{C}_i^*} (\mathbf{E} \circ \tilde{K})(\mathbf{d}, \mathbf{p}_j)}{\sum_{\mathbf{p}_j \in \mathscr{C}_i^*} \alpha_{\tilde{K}}(\mathbf{d}, \mathbf{p}_j)} \tag{3.12}$$

Où $\alpha_{\tilde{K}}(\mathbf{d}, \mathbf{p}_j)$ est l'opacité de la contribution retournée par l'évaluation d'un noyau $\tilde{K}$. Le problème posé par cette accumulation est qu'aucune information de profondeur n'est prise en compte, ne permettant pas de produire une atténuation correcte si des variations de couleurs existent entre les échantillons. Les informations de profondeur étant cruciales pour produire un résultat convaincant, nous utilisons une fonction de *Transparence Indépendante de l'Ordre* (ou **OIT** pour *Order Independent Transparency*). Le principe élémentaire derrière cette fonction d'**OIT** est de définir un poids selon la profondeur du noyau ou de l'échantillon dans l'écran, donné dans notre cas par la fonction $T(\mathbf{p}_j)$. La

fonction d'**OIT** ci-après est calculée sur la profondeur de l'échantillon le plus contributif du noyau évalué selon la direction du rayon dans la cellule, pour approximer efficacement l'atténuation entre les noyaux. En introduisant cette fonction dans l'équation 3.12, celle-ci devient :

$$L_{\mathscr{C}_i} = \frac{\sum_{\mathbf{p}_j \in \mathscr{C}_i^*} T(\mathbf{p}_j)(\mathbf{E} \circ \tilde{K})(\mathbf{d}, \mathbf{p}_j)}{\sum_{\mathbf{p}_j \in \mathscr{C}_i^*} T(\mathbf{p}_j)\alpha_{\tilde{K}}(\mathbf{d}, \mathbf{p}_j)} \tag{3.13}$$

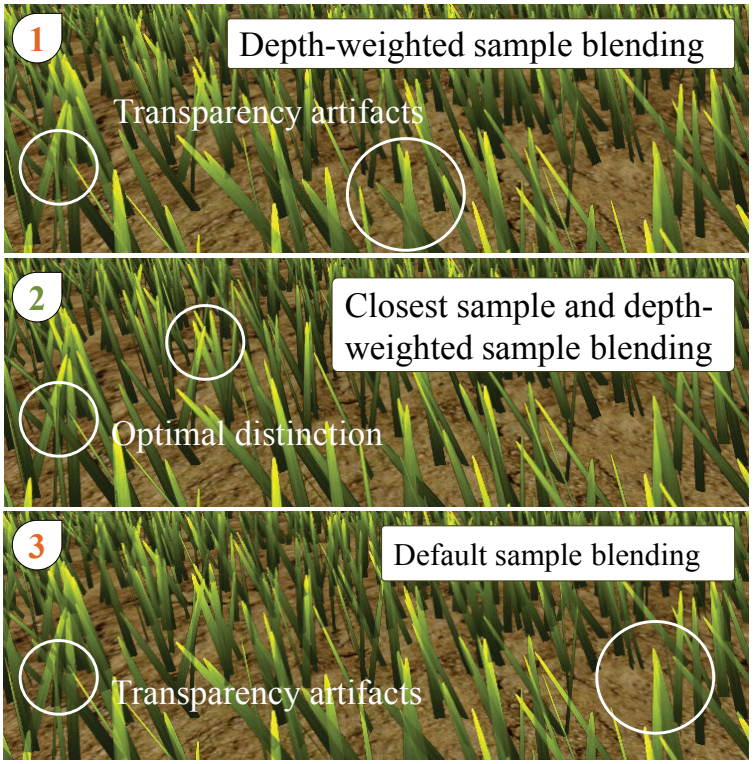


FIGURE 3.5 – Impact de l'*Order Independent Transparency*.
1re ligne: une fonction de pondération selon la profondeur telle que décrite par McGuire et Bavoil [MB13] est utilisée. 2e ligne: mélange de l'échantillon contributif le plus proche avec l'accumulateur d'échantillons pondérés selon la profondeur 3e ligne : aucune méthode d'**OIT** utilisée (tous les noyaux sont mélangés, équivalant à l'utilisation d'une fonction T constante).

La fonction $T(\mathbf{p}_j)$ peut être définie de différentes manières. Nous avons tout d'abord expérimenté les formulations génériques proposées par McGuire et Bavoil [MB13]. Une amélioration de la distinction des noyaux est remarquable en utilisant leurs formulations (figure 3.5.1), comparativement aux résultats sans **OIT** (figure 3.5.3). Cette dernière configuration correspond à $T(\mathbf{p}_j) = 1, \forall j$. Certains noyaux opaques, en particulier ceux situés près les uns des autres, ne sont pas correctement rendus. L'erreur de transparence dans le cas des parties opaques des noyaux proches de la caméra est liée au mélange de leur contribution malgré une opacité maximale atteinte. L'approche par échantillonnage du rayon permet une correction rapide de ce problème : nous proposons de conserver, lors de la distribution et de l'évaluation des noyaux dans une cellule, la contribution la plus proche de la caméra dans la cellule. Cette contribution est ensuite mélangée aux contri-

butions pondérées par **OIT** :

$$L_{\mathscr{C}_i} = \tilde{K}(\mathbf{d}, \mathbf{p}_f)\alpha_f + \left(\frac{\sum_{\mathbf{p}_j \in \mathscr{C}_i^*} T(\mathbf{p}_j)(\mathbf{E} \circ \tilde{K})(\mathbf{d}, \mathbf{p}_j)}{\sum_{\mathbf{p}_j \in \mathscr{C}_i^*} T(\mathbf{p}_j)\alpha_{\tilde{K}}(\mathbf{d}, \mathbf{p}_j)} \right) (1 - \alpha_f) \tag{3.14}$$

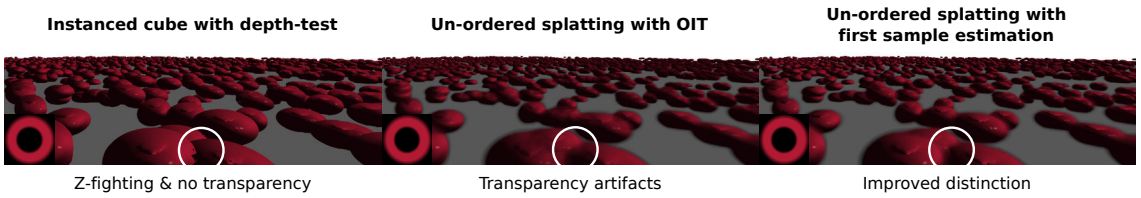
$\mathbf{p}_f$ et α_f sont ici la position et la transparence du noyau correspondant à l'échantillon contributif le plus proche de la caméra dans la cellule évaluée. Pour l'évaluation de l'équation 3.8, l'opacité de la cellule est évaluée séparément par une accumulation non pondérée de l'opacité des contributions des noyaux. Un résultat calculé par cette amélioration est presque identique dans les cas des noyaux quasi transparents mais une amélioration significative du résultat peut être constatée lorsque des noyaux opaques sont utilisés (figure 3.5.2).

Cet échantillonnage particulier permet d'évaluer notre modèle plus efficacement qu'un échantillonnage pas à pas standard du rayon, et des effets supplémentaires tels que l'ombrage de la surface ou encore l'auto-ombrage des noyaux pourraient également être évalués, en lançant des rayons supplémentaires pour chaque noyau. Mais les évaluations supplémentaires de ces effets requièrent une puissance de calcul non négligeable pour que le rendu reste interactif, les performances de notre échantillonnage étant déjà grandement limitées par le parcours de la grille : le parcours des cellules successives par l'algorithme de Bresenham, dans lesquelles sont réalisées à chaque itération la distribution et l'évaluation des noyaux, implique une forte composante itérative dans l'évaluation. Cette méthode ne profite également de la puissance de la carte graphique que de manière limitée : l'intégralité des calculs est réalisée dans notre implantation par un *fragment shader*, utilisant majoritairement les unités de traitement par pixel. Cette approche utilise très peu les unités de traitement de géométrie, créant un déséquilibre de charge sur la carte graphique et limitant rapidement la vitesse de rendu avec l'augmentation du nombre de cellules traversées. Pour maximiser les performances d'évaluation, il est possible d'utiliser des approches de rendu volumétrique basées objet, telles que le *EWA volume splatting*.

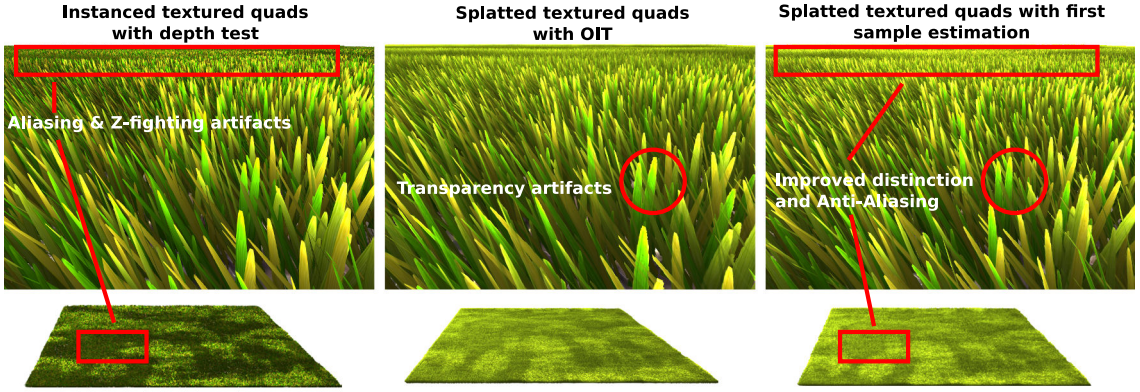
3.3.4 Rendu par projection de noyaux instanciés

L'*EWA volume splatting* de Zwicker *et al.* [ZPvBG01] propose un schéma alternatif se basant sur la rastérisation des noyaux K individuels. En pratique, les gaussiennes composant les noyaux sont découpées en plusieurs gaussiennes plates faisant face à la caméra, ces coupes étant ensuite rastérisées directement en espace écran. Cette technique est cependant utilisable uniquement pour un faible nombre de gaussiennes ou si une évaluation correcte de l'atténuation n'est pas requise : l'*EWA volume splatting* requiert un tri des gaussiennes plates selon la profondeur pour l'évaluation de l'atténuation. Ce tri réduit fortement les performances dans le cas d'un grand nombre de gaussiennes, ce qui est notre cas.

Le *depth-peeling* de Nagy et Klein [NK03] peut être utilisé pour accélérer le tri, mais cette technique reste coûteuse en temps de calcul pour un grand nombre de gaussiennes. Chaque noyau en recouvrant un autre à l'écran nécessiterait une passe de *depth-peeling*.



(a) Rendu de noyaux volumiques



(b) Rendu de noyaux plats

FIGURE 3.6 – Comparaison des méthodes de rendu d'ordres objet
Le résultat calculé par instanciation de noyaux avec test de profondeur (1re colonne) est efficace en termes de performances et de qualité pour les éléments proches de la caméra dans le cas des noyaux plats, mais ne gère pas correctement la transparence. Des artefacts de scintillement et d'aliassage apparaissent sur les parties lointaines et sur les recoupements de noyaux volumiques. Ces artefacts pourraient être corrigés par un schéma de filtrage complexe. La projection de noyaux pondérés par une fonction d'**OIT** (2e colonne) ne présente pas ces artefacts mais des artefacts de transparence sont visibles à proximité de la caméra. Notre pipeline de rendu (3e colonne) corrige ces problèmes en utilisant les informations de profondeur fournies par l'instanciation.

Pour approximer l'atténuation à un coût nettement inférieur, nous proposons une autre approche : une accumulation des contributions non ordonnées des noyaux pondérées selon la profondeur dans la vue lors du rendu. Le résultat final est obtenu par normalisation dans une passe de recomposition de l'image. La contribution de chaque noyau volumique est approximée par pixel par une découpe ordonnée selon la profondeur. En pratique, la boîte englobante de chaque noyau est rastérisée pour créer les rayons de vue traversant le noyau, et cette découpe est implicitement obtenue par l'échantillonnage du rayon lancé à travers le noyau K (équation 3.11).

Nous utilisons une fonction d'**OIT** définie cette fois non pas par cellule, mais pour l'ensemble de l'espace d'évaluation. Similairement à l'échantillonnage précédent de la cellule, les contributions de tous les noyaux sont également accumulées sans tri de ces derniers selon la profondeur ou sans utiliser de *depth-peeling*, afin d'éviter le surcoût de ces pré-traitement sur la passe de rendu. La fonction d'**OIT** $T(\mathbf{p}_j)$ ci-après définit une pondération selon la profondeur dans l'écran des échantillons évalués. Ce changement d'espace d'application de l'**OIT** signifie également que la normalisation du résultat n'est plus réalisée par cellule, mais sur le résultat final obtenu. En utilisant notre combinaison

de l'*EWA splatting* et de cette fonction d'**OIT** , l'équation 3.10 peut être remplacée par :

$$L_{\mathscr{C}_i} \approx \sum_{\mathbf{p}_j \in \mathscr{C}_i^*} T(\mathbf{p}_j)\tilde{K}(\mathbf{d}, \mathbf{p}_j) \tag{3.15}$$

Une normalisation est nécessaire pour obtenir une approximation de L, le résultat étant lié à une accumulation pondérée par la fonction $T(\mathbf{p}_j)$, cette accumulation dépendant elle-même du nombre d'échantillons et de cellules :

$$L \approx \frac{\sum_{i=0}^{N-1} L_{\mathscr{C}_i}}{\sum_{i=0}^{N-1} \sum_{\mathbf{p}_j \in \mathscr{C}_i^*} T(\mathbf{p}_j)} \tag{3.16}$$

Un constat similaire à celui fait pour l'échantillonnage pas à pas du rayon est notable : bien que la distinction entre les éléments soit améliorée, des artéfacts de transparence sont visibles sur les noyaux opaques proches de la caméra (figure 3.6 au centre). Pour un bruit de spot composé majoritairement de noyaux opaques, le résultat visuel peut être significativement amélioré en maximisant la contribution à la couleur finale L du noyau le plus proche de la caméra. Dans un pipeline de rendu, ce noyau peut être aisément déterminé en utilisant un test de profondeur. L est alors obtenu en combinant l'approximation précédente basée sur l'**OIT** et l'*EWA splatting* avec la contribution de l'échantillon le plus proche :

$$L \approx \tilde{K}(\mathbf{d}, \mathbf{p}_f)\alpha_f + \left(\frac{\sum_{i=0}^{N-1} L_{\mathscr{C}_i}}{\sum_{i=0}^{N-1} \sum_{\mathbf{p}_j \in \mathscr{C}_i^*} T(\mathbf{p}_j)} \right) (1 - \alpha_f) \tag{3.17}$$

$\mathbf{p}_f$ est l'impulsion correspondant au noyau le plus proche de la caméra pour le rayon évalué, et α_f est sa transparence.

En pratique, nos expérimentations montrent qu'un résultat calculé par cette amélioration est presque identique au résultat rendu par projection de noyaux avec une fonction d'**OIT** standard dans le cas de noyaux quasi transparents, mais elles montrent également une amélioration significative du résultat pour des noyaux ayant une partie opaque (figure 3.6 à droite). Le résultat est visuellement très proche, voire identique à celui obtenu par notre méthode de rendu par échantillonnage pas à pas de la grille volumique. Par rapport à cette dernière, les performances sont nettement supérieures lorsque la densité d'éléments à l'écran est relativement faible (< 1 million). Mais l'échantillonnage pas à pas de la grille conserve l'avantage sur les densités extrêmes, où les méthodes d'ordre objet évaluent énormément d'échantillons non contributifs car cachés par des éléments plus proches de la caméra.

3.4 Résultats

Les figures 3.7, 3.8, 3.11 et 3.13 présentent des exemples de noyaux plats opaques, et les figures 3.12, 3.14 et 3.15 présentent des exemples de noyaux volumiques distribués sur des objets 3D. Les images rendues par projection de noyaux sont évaluées dans une fenêtre de résolution 1920×1080 pixels et les images rendues par échantillonnage pas à pas de la grille volumique sont évaluées dans une fenêtre de résolution 1024×1024 pixels. Ces images sont générées interactivement par une GeForce GTX 980 et un moteur de rendu basé sur OpenGL 4.3.

3.4.1 Performances

Tous les exemples utilisant des gaussiennes plates évaluées par projection sont générés entre 15 et 200 millisecondes, selon la densité de noyaux à l'écran et le point de vue. Les figures 3.7 et 3.8 sont respectivement générées en ≈ 300 millisecondes et ≈ 650 millisecondes par l'échantillonnage pas à pas de la grille volumique. Pour simuler l'auto-ombrage de la fourrure et de l'herbe, nous utilisons le modèle d'occultation volumique de Neulander *et al.* [NKB13].

Les performances de nos méthodes de rendu sont fortement dépendantes de la densité de noyaux affichés à l'écran et de la complexité de ces noyaux composant le bruit de surcouche volumique. La figure 3.6 présente différentes méthodes de rendu basées objet pouvant être utilisées pour évaluer notre modèle de bruit. Le rendu de noyaux opaques par instanciation de géométrie offre les meilleures performances (de 60 images par seconde pour 800000 noyaux jusqu'à 3 images par seconde pour 10 millions de noyaux) pour un nombre limité d'éléments (courbe jaune de la figure 3.9 et courbes bleues de la figure 3.10). Mais cette méthode introduit également de nombreux artefacts d'aliassage (figure 3.6 à gauche) avec la diminution de la taille des noyaux, due à la limite de précision du tampon de profondeur, et ne gère pas correctement la semi-transparence. Les artefacts d'aliassage peuvent être évités et la semi-transparence peut être correctement traitée si une méthode telle que l'*EWA splatting* est utilisée. L'*EWA splatting* standard n'est cependant pas utilisable en temps réel pour la quantité d'éléments rendus dans notre cas, le tri selon la profondeur (ou un *depth peeling*) impactant significativement les performances et le coût en stockage (courbe orange de la figure 3.9). L'*Order Independent Transparency* offre une alternative rapide à ce tri, permettant d'atteindre des performances proches de celles obtenues avec l'instanciation de géométrie (courbes orange de la figure 3.10), mais des artefacts de transparence peuvent apparaître et être visibles pour les éléments proches de la caméra. Une fonction d'**OIT** basique n'est pas assez précis pour une distinction efficace des noyaux très proches les uns des autres. Nous évitons ces artefacts de transparence en conservant l'échantillon contributif le plus proche de la caméra. Cette contribution est obtenue dans le cas du rendu par projection de noyaux grâce à une passe d'instanciation de géométrie en supplément de la passe de mélange des contributions pondérées par une fonction d'**OIT**, au prix d'une perte de performances de l'ordre de la moitié de la vitesse de rendu des méthodes précédentes (courbes jaunes de la figure 3.10).

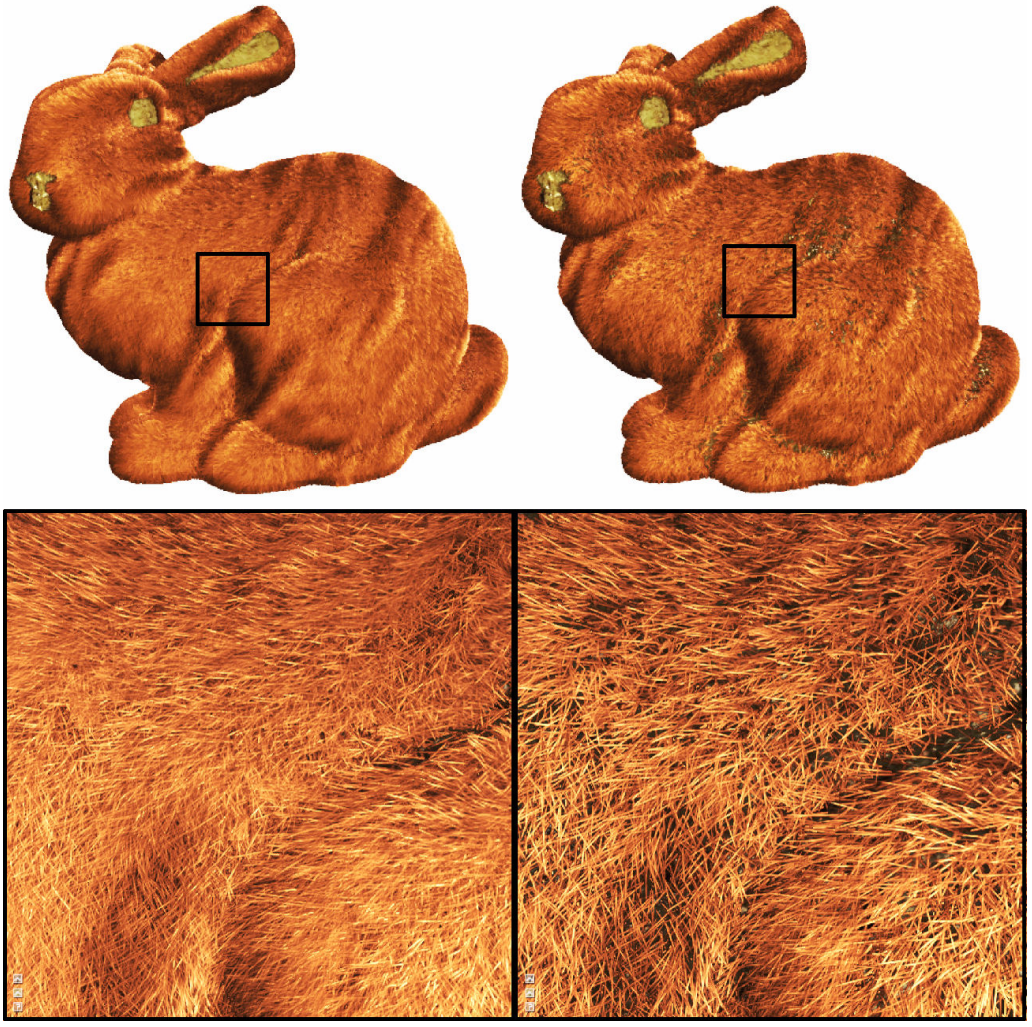
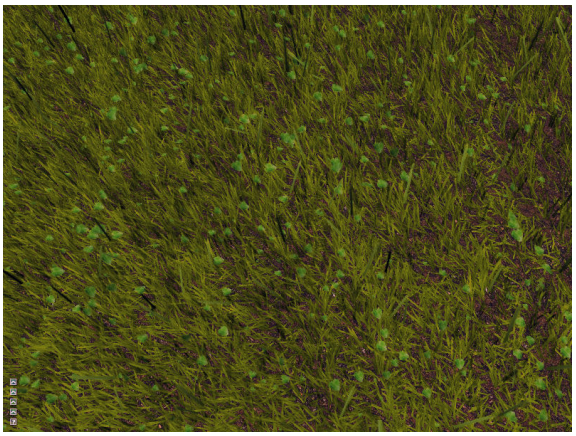


FIGURE 3.7 – Un exemple de texture de fourrure rendu par notre méthode d'échantillonnage pas à pas de la grille volumique.
L'apparence de ces modèles est définie en utilisant ≈ 20 millions de petits noyaux pour le lapin de gauche et ≈ 8 millions de petits noyaux pour le lapin de droite. Ces résultats sont rendus en ≈ 300 millisecondes.

Ces différentes méthodes se trouvent cependant rapidement submergées lorsque la densité d'éléments est extrême, le nombre d'éléments rasterisés devenant trop important pour la carte graphique. Notre méthode d'échantillonnage du rayon devient dans ce cas plus efficace : cette densité importante provoque un arrêt rapide de l'évaluation du rayon, limitant la perte de performances liée au rendu d'éléments cachés dans le cas des méthodes de rendu basées objet. Bien qu'inadaptée aux applications temps réel, notre implantation de cette méthode dans un *fragment shader* conserve la possibilité d'une édition interactive, et ce même pour une densité extrême d'éléments affichés (courbe bleue de la figure 3.9).

L'utilisation d'une définition du noyau basée sur une somme de gaussiennes ellipsoïdales nous permet de modéliser aussi bien des surfaces fines, telles que des brins d'herbe, que des éléments complexes. La figure 3.12 présente des exemples de textures de surcouche procédurales obtenues à l'aide de plusieurs gaussiennes. Cette image présente,



(a) Motif de surcouche de végétation

(b) Motif appliqué dans une scène

FIGURE 3.8 – Un exemple de surcouche de végétation rendu par notre méthode d'échantillonnage pas à pas de la grille volumique.
Le motif (a) est composé de différents noyaux texturés pour représenter différents éléments de végétation (des brins d'herbe et des feuilles). La scène (b) est composée de $\approx 2,5$ millions de brins d'herbes et 60000 de feuilles. Des cartes d'orientation et de densité sont peintes interactivement par l'utilisateur.
Le résultat est rendu en ≈ 650 millisecondes.

de haut en bas, différentes distributions d'impulsions allant d'une distribution complètement aléatoire à une distribution semi-régulière. Ces textures ont été modélisées interactivement en modifiant les gaussiennes de k (l'anisotropie, la position et l'orientation) et en fournissant différents profils de densité périodique (les vignettes en bas à gauche). Cette figure démontre qu'un nombre faible de gaussiennes peut être suffisant pour modéliser des textures aléatoires complexes. Des textures de surcouche encore plus complexes peuvent être obtenues en contrôlant la courbure, l'autorotation, la taille et la couleur des noyaux, ainsi qu'en utilisant différents types de noyaux et différentes distributions.

La figure 3.15 présente des exemples de noyaux volumiques 3D distribués sur des objets 3D. Tous les détails de ces objets sont créés par la combinaison de noyaux volumiques semi-transparents. Similairement aux exemples précédents, les noyaux peuvent être édités interactivement et contrôlés en utilisant des fonctions ou des images basse résolution. La performance de nos méthodes de rendu dans le cas des noyaux 3D semi-transparents

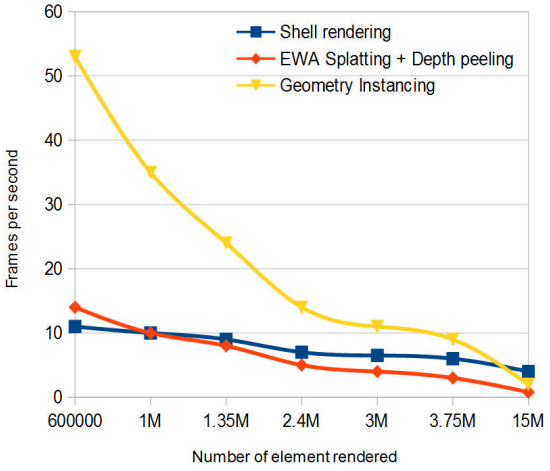
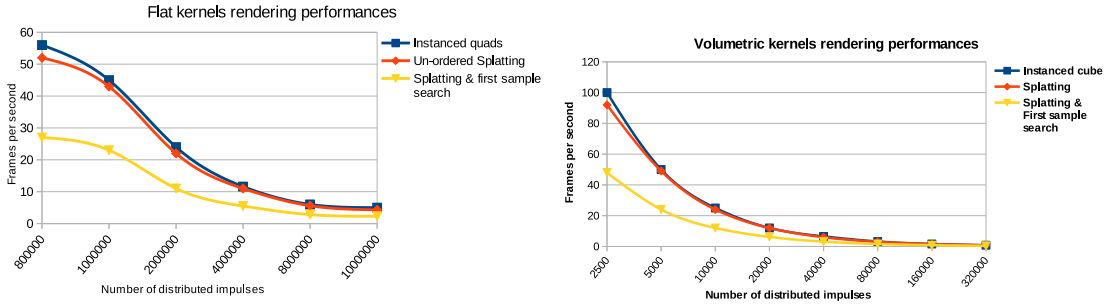


FIGURE 3.9 – Comparatif de performances entre l'échantillonnage des rayons, l'instanciation de géométrie et la projection de noyaux.
Pour un nombre de brins réduit, l'instanciation de géométrie et la projection de noyau associé à un *depth peeling* restent plus efficaces, mais l'ordonnancement n'est que faiblement approximé par le *depth peeling*, et de nombreux artéfacts de scintillement apparaissent avec l'instanciation de géométrie. Bien que moins performante pour un nombre réduit d'éléments, notre méthode d'échantillonnage des rayons de vue (*shell rendering* ci-dessus) offre une meilleure qualité de résultat et peut supporter un plus grand nombre d'éléments rendus simultanément.



(a) Rendu d'éléments plats. (b) Rendu d'éléments volumiques ($N_s = 32$).

FIGURE 3.10 – Comparatif de performances des méthodes de rendu d'ordre objet sans ordonnancement. Les performances de l'instanciation d'objets et de la projection de noyaux non ordonnés sont similaires mais ces méthodes laissent apparaître de nombreux artéfacts. Au prix d'une division par deux de la vitesse de rendu, notre méthode permet un rendu de bien meilleure qualité.

est fortement dépendante à la fois du nombre de noyaux et du nombre de coupes N_s de l'équation 3.11 (et par extension de la précision de l'évaluation du noyau). Dans le cas de notre méthode de rendu par projection de noyaux, les performances vont de 55 images par seconde pour 2500 noyaux volumiques jusqu'à $1,5$ images par seconde pour 300000 noyaux.

3.4.2 Occupation mémoire

L'empreinte mémoire de notre modèle est extrêmement faible, la définition du noyau et celle de la distribution ne demandant elles-mêmes que quelques kilo-octets. Cette empreinte ne dépend pas du montant total de noyaux rendus, mais principalement du

nombre de différents paramétrages de noyaux utilisés. Les paramètres régissant l'apparence du noyau tels que l'orientation, la distribution, la densité et la taille peuvent être définies sous la forme d'images basse résolution peintes sur la surface, ou être calculés procéduralement.

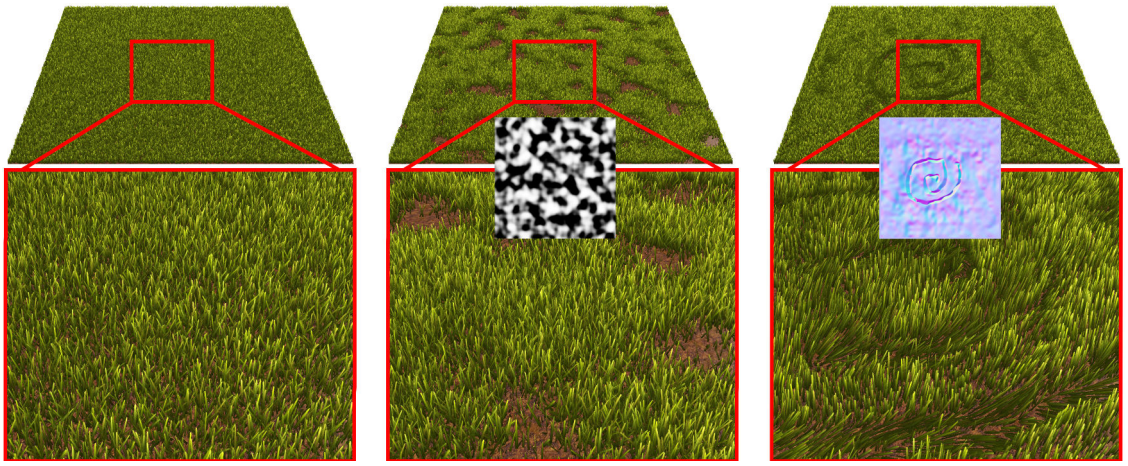


FIGURE 3.11 – Contrôles des paramètres locaux de la texture procédurale.
(À gauche) le résultat utilisant une distribution uniforme, (au centre) densité contrôlée par l'utilisateur, (à droite) orientation contrôlée par l'utilisateur. Ces différents contrôles sont directement peints sur la surface.

La figure 3.11 présente certains effets visuels du contrôle utilisateur sur le bruit. Celui-ci peut être édité et animé interactivement en modifiant la densité et l'orientation des noyaux. Cela est possible grâce à l'évaluation à la volée des différents paramètres, sans aucune forme de précalcul nécessaire (contrairement aux méthodes basées sur le préfiltrage de voxels). La modification des paramètres peut être exprimée à travers une fonction temporelle (telle qu'une fonction de vent), ou une combinaison de textures basse résolution. Ces dernières permettent à l'utilisateur de contrôler efficacement les noyaux et facilitent l'édition de caractéristiques locales dans la texture procédurale.

Comme présenté dans les figures 3.7 et 3.13, nos méthodes de rendu permettent des prises de vue rapprochées dans lesquelles la microgéométrie complexe devient plus précisément visible (les brins de fourrure sont plus facilement distingués individuellement). Vus de loin, ces brins individuels sont rendus sans aliassage par notre méthode de rendu par projection de noyaux intégrant une fonction d'**OIT** et par notre échantillonnage des rayons de vue grâce à l'utilisation de fonctions gaussiennes filtrées.

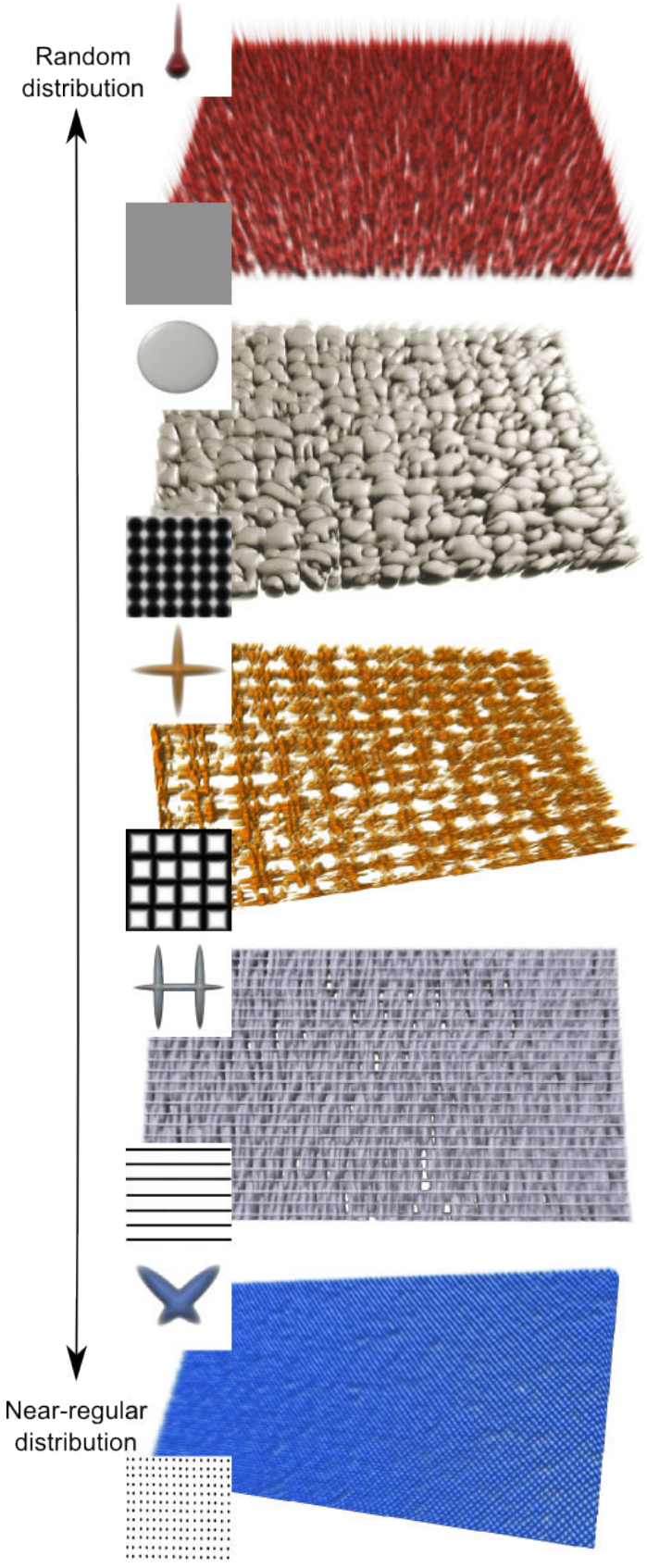


FIGURE 3.12 – Textures volumiques définies par un bruit de surcouche non uniforme de surcouche.
En haut à gauche : une somme de gaussiennes elliptiques est utilisée pour définir le noyau k (jusqu'à 3 gaussiennes dans cet exemple). En bas à gauche : le champ scalaire contrôlant la distribution d'impulsions. Les valeurs proches du noir impliquent une densité d'impulsions plus importante. Toutes les textures produites sont extrêmement compactes (seuls la fonction de bruit, une matrice 4×4 par gaussienne et un champ de densités sont stockés dans ces exemples), continues et apériodiques.

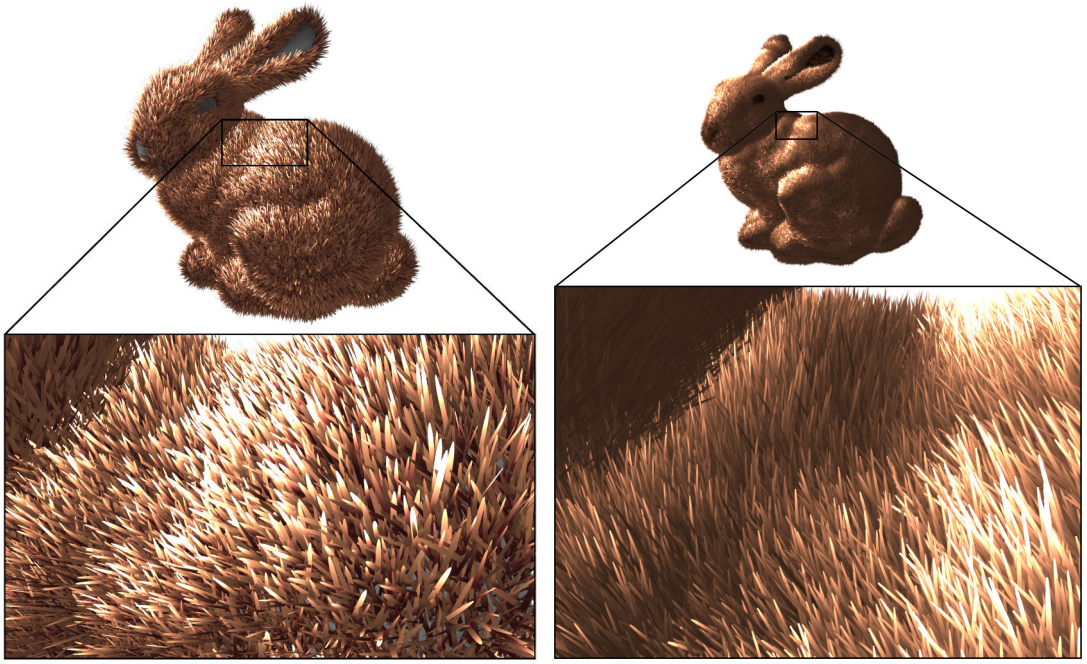


FIGURE 3.13 – Un exemple de texture de fourrure rendue par notre méthode de projection de noyaux instanciés. L'apparence de ces modèles est définie en utilisant ≈ 400000 petits noyaux pour le lapin de gauche (10 images par seconde) et $\approx 1,6$ millions de petits noyaux pour le lapin de droite (3 images par seconde). Chacun de ces noyaux peut être édité interactivement.

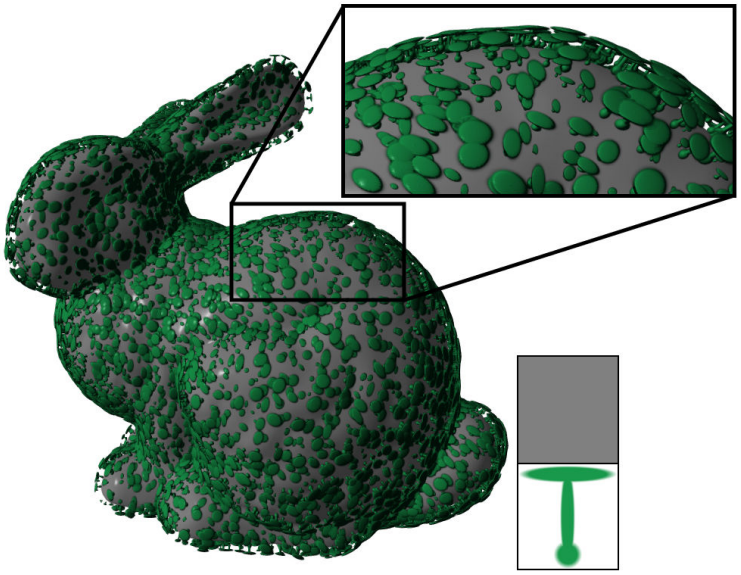


FIGURE 3.14 – Un exemple de motif volumique appliqué sur un objet et rendu par projection des noyaux. L'apparence de la texture de surcouche est rendu en 300 millisecondes pour ≈ 20000 noyaux distribués. Le paramétrage du noyau est défini par 3 gaussiennes (vignette en bas à gauche) et un profil de distribution aléatoire (vignette grise en bas à gauche). Les noyaux distribués subissent également des variations aléatoires de tailles selon les axes (u, v) de la surcouche, visibles sur le résultat proche de la caméra.

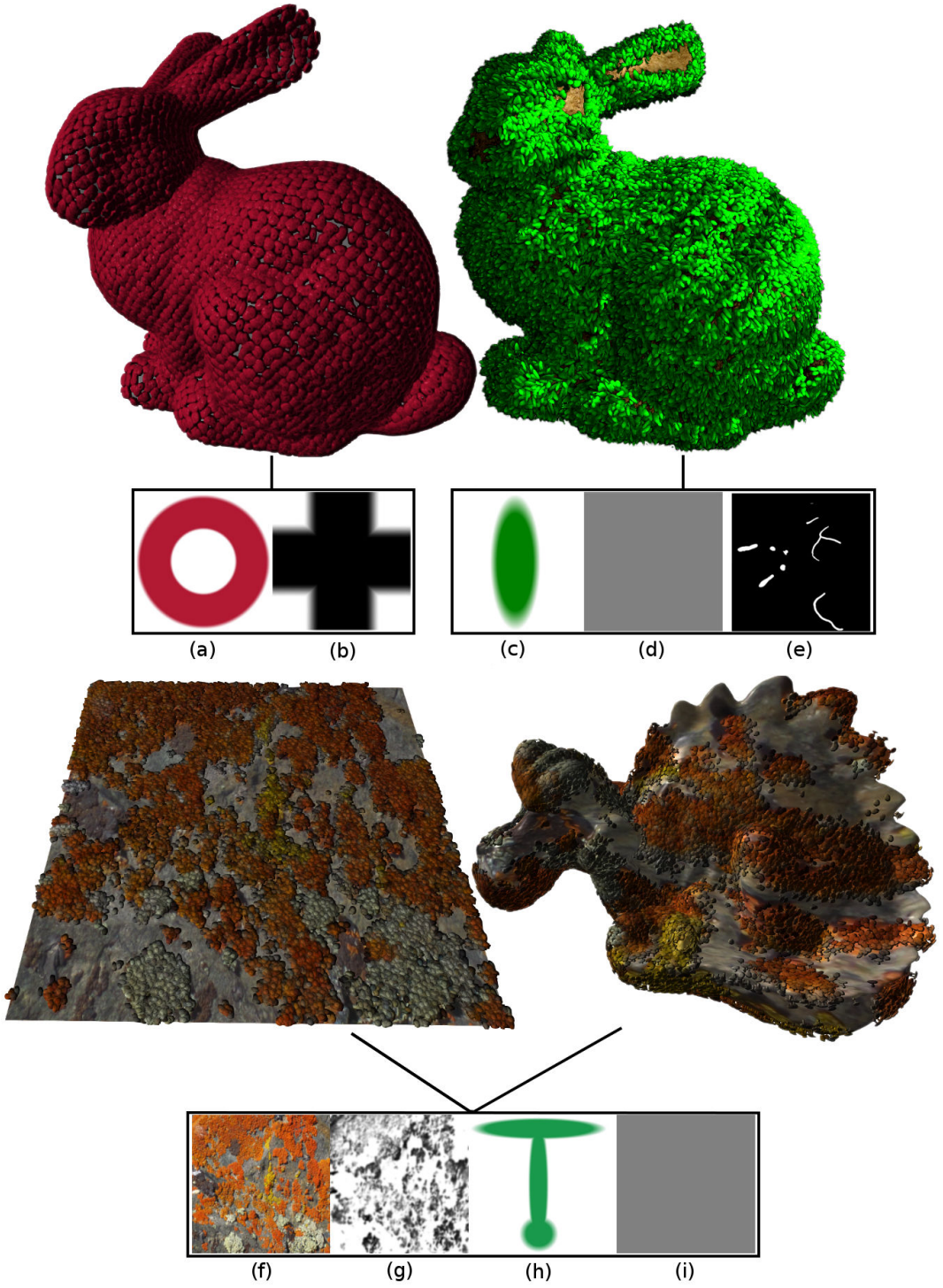


FIGURE 3.15 – Motifs volumiques appliqués sur différents modèles 3D.
Le motif appliqué sur le premier lapin de gauche utilise un profil de noyau en forme d'anneau (a) composé de 2 gaussiennes et un profil permettant de créer une distribution semi-régulière (b). Le second lapin utilise un unique noyau gaussien (c) associé à un profil de distribution aléatoire (d), une carte de densité (e) est également utilisée pour limiter la distribution sur des zones spécifiques. Les deux lapins utilisent ≈ 50000 impulsions et sont rendus en 500 millisecondes par projection de noyaux. Le motif de lichen appliqué sur le dragon et le plan est créé en utilisant une carte de couleur (f), une carte de densité (g), un noyau composé de 3 gaussiennes (h) et une distribution aléatoire (i). Ces deux derniers résultats utilisent ≈ 400000 impulsions et sont rendu en 650 millisecondes par projection de noyaux.
Tous les résultats utilisent 32 échantillons par rayons pour chaque noyau évalué.

3.5 Conclusion

Nous avons présenté dans ce chapitre une extension du bruit de spot localement contrôlé pour l'édition interactive de textures de surcouche volumiques. Cette extension peut produire de nombreuses textures volumiques procédurales différentes, allant de motifs 3D quasi réguliers jusqu'à des motifs complètement aléatoires, et ce pour une empreinte mémoire minimale. Notre évaluation optimisée du noyau, utilisée dans les différentes méthodes de rendu présentées, permet la visualisation et l'édition interactive de motifs de surcouche volumiques complexes. L'interactivité est garantie même pour plusieurs millions de noyaux, améliorant significativement l'édition de ces textures.

Notre méthode de rendu par projection de noyaux instanciés et pondérés par une fonction d'**OIT**, bien qu'efficace, perd une portion significative de ces performances à cause du test de profondeur utilisé pour l'évaluation de l'échantillon significatif le plus proche : le test est réalisé lors d'une deuxième passe de rendu pour laquelle chaque noyau est instancié et rastérisé une nouvelle fois. L'évolution des contrôleurs mémoire présents dans les cartes graphiques permet, pour les génération compatibles OpenGL 4.3, d'utiliser des opérations atomiques sur certains emplacements critiques de la mémoire. Une possible amélioration, selon l'impact de ces opérations sur les performances, serait de réaliser la recherche de l'échantillon le plus proche, ou l'opération d'accumulation d'échantillons pondérés, en utilisant un tampon de profondeur ou un tampon de contribution modifié par ces opérations atomiques dans une unique passe de rendu. Cette amélioration ne corrige pas cependant un des problèmes majeurs du rendu par projection d'éléments volumiques : la projection et la rastérisation de l'ensemble noyaux entraînent une limitation du nombre maximal d'éléments rendus, due à la limite de traitement de primitives géométriques ou de fragments simultanés de la carte graphique.

Parmi les extensions possibles de ce modèle de bruits volumiques et de ces méthodes de rendu, une amélioration des performances d'évaluation du modèle pourrait être obtenue par l'utilisation d'un schéma de gestion du niveau de détails pour la sélection d'un noyau et/ou d'une méthode de rendu plus adaptée en fonction de la distance à la caméra. L'utilisation de différents noyaux décrivant un même ensemble de détails mais à différents niveau d'échelle permettrait de réduire la complexité d'évaluation de la surcouche. Une autre solution possible pourrait être d'adapter l'évaluation du motif selon les conditions de visualisation, en utilisant par exemple de la tessellation et de l'instanciation pour les objets proches de la caméra et une approximation basée sur un filtrage statistique du modèle pour la visualisation des éléments lointains. Nous pensons qu'un tel schéma de rendu multi-échelles permettrait à terme un rendu temps réel réaliste de textures de surcouche volumiques procédurales.

Conclusion et Perspectives

Au cours de cette thèse, nous avons proposé une approche statistique pour la modélisation interactive de motifs procéduraux volumiques naturels, allant de stochastiques à semi-réguliers et incluant des éléments structurels. Nous avons montré que ce type de motifs, et en particulier ceux formés par des petits objets de surface quasi similaires tels que les champs d'herbe ou la fourrure, posent de nombreux problèmes de modélisation et de rendu : le nombre d'objets similaires mais non identiques composant ces motifs les rend à la fois long et fastidieux à éditer sans une méthode de génération adaptée, mais également complexes à visualiser sans une représentation adaptée au niveau de détails voulu dans l'image. Nous avons décidé d'utiliser un bruit procédural pour générer à la volée ces détails directement lors de l'évaluation d'une surcouche volumique. Les bruits procéduraux présentés dans la partie 1.3 sont une solution possible pour produire des motifs stochastiques continus, et parmi eux le bruit local à phase aléatoire permet également reproduire des éléments structurels. Mais ces bruits ne peuvent créer des motifs semi-réguliers. La corrélation non intuitive entre l'aspect visuel du motif produit et l'espace fréquentiel, dans lequel interviennent les paramètres du bruit, complexifie également l'édition interactive pour un artiste.

Notre premier objectif était de pouvoir, à l'aide d'un bruit, créer un motif semi-aléatoire comprenant des éléments structurels pouvant être répartis aléatoirement ou régulièrement dans l'espace. Nous avons pour cela développé un nouveau modèle de bruit facilitant l'édition spatiale de motifs. Ce modèle est basé sur l'utilisation de noyaux de convolution construit par une somme de fonctions gaussiennes elliptiques. Ces noyaux permettent de définir spatialement des éléments structurels dans des motifs aléatoires. Cette fonction gaussienne peut être utilisée par un artiste comme une primitive de modélisation des éléments du motif, et permet d'utiliser des méthodes d'extraction automatique de ces éléments depuis un exemple. Nous avons également introduit une distribution non uniforme d'impulsions permettant de créer des motifs semi-réguliers. Les noyaux et la distribution peuvent être localement contrôlés pour créer des motifs complexes comprenant différents niveaux de structures, tels que des motifs de tissu quasi réguliers intégrant des micro variations.

Notre second objectif était ensuite de pouvoir créer et visualiser en temps réel des détails de surface 3D tels que des brins d'herbe et des poils à l'aide de ce bruit. Nous avons proposé une extension du modèle de bruit précédent pour la modélisation interactive de motifs de surcouche volumiques anti-aliasés. Pour permettre cette évaluation anti-aliasée,

le modèle étendu intègre une formulation filtrable analytiquement du noyau de convolution précédent, associée à une méthode d'évaluation de la contribution de ce noyau filtré à l'image. Cette dernière est utilisée à travers deux méthodes de rendu implantées sur carte graphique et optimisées pour l'évaluation de notre bruit procédural. Une première méthode d'ordre image, basée sur un échantillonnage pas à pas d'une grille volumique régulière englobant la surface, est implantée dans un *fragment shader* et permet un rendu interactif de haute qualité du résultat, et ce même pour une quantité extrême de détails distribués sur la surface. Une deuxième méthode d'ordre objet, basée sur l'instanciation et la projection de l'ensemble des noyaux, permet un rendu plus rapide de notre modèle, en conservant une qualité quasi-équivalente, mais pour une quantité maximale d'éléments affichés moindre. À l'aide de ce modèle de bruit et de ces méthodes de rendu, nous avons pu créer et visualiser interactivement des champs d'herbe et des fourrures très denses, mais également des objets volumiques plus complexes distribués sur des surfaces. Nos méthodes de rendu ne permettent pas une évaluation suffisament rapide pour les applications temps réel, mais les performances de notre dernière méthode se rapprochent cependant de cet objectif.

Perspectives

Notre modèle de bruit et nos méthodes de rendu permettent effectivement l'édition spatiale et la visualisation interactive de détails 2D et 3D de surface. Mais de nombreuses améliorations sont possibles, tant en termes de modélisation qu'en termes de performances. Nous présentons ci-après diverses perspectives d'améliorations pouvant combler les lacunes et étendre les possibilités offertes par le modèle de bruit et par les méthodes de rendu présentées dans cette thèse.

Gestion de l'échelle des détails

L'un des problèmes posés par notre modélisation est qu'un même noyau, bien qu'il soit filtré, est utilisé quelle que soit l'échelle du détail qu'il représente dans l'image. Nous avons vu dans la section 1.1.4 qu'il est possible d'optimiser la complexité du rendu en sélectionnant une représentation adéquate des détails de surface en fonction de la distance de ces détails à la caméra. Une approche similaire serait applicable dans notre modèle, en utilisant différentes configurations de noyau représentant un même ensemble de détails selon différents niveaux d'échelle dans l'image. La définition de ces différents noyaux, et de possibles noyaux de transitions pour limiter les discontinuités visuelles lors de l'évaluation, est cependant un problème complexe. Bien qu'il existe des méthodes pouvant potentiellement générer des simplifications d'un noyau (telle que la décomposition en ondelettes évoquée dans la section 2.5), une multiplication des différents paramétrages pourrait être nécessaire pour un résultat ne laissant pas apparaître les transitions de configuration.

Les deux méthodes de rendu présentées dans le chapitre 3 introduisent une autre possibilité pour la gestion du niveau de détails. Dans le cas des détails volumiques, en utilisant une unique configuration de noyau, différentes méthodes de rendu de la surcouche pourraient être utilisées selon la distance de la surcouche à la caméra. Comme précisé dans la section 1.1.4, les méthodes d'ordre objet utilisées pour le rendu de géométrie sont très efficaces pour le rendu d'objets proche de la caméra. Lorsque une cellule est loin de la caméra, le filtrage analytique de notre formulation a pour effet indirect qu'un plus grand nombre des noyaux de la cellule contribue au rayon. Cet aspect favorise les méthodes d'ordre image, le rayon atteignant dans ce cas plus rapidement l'opacité maximale.

À terme, l'utilisation de noyaux intégrant de manière continue différentes échelles de visualisation, associée à différentes méthodes de rendu selon l'échelle visuelle du détails, permettrait d'optimiser finement la complexité et d'atteindre un niveau de performance suffisant pour les applications temps réel.

Reproduction de dimensions supérieures

Notre modèle de bruit n'est pas restreint par une dimension particulière, mais la méthode d'extraction automatique de paramètres proposée dans le chapitre 2 est limitée à la seule analyse de motifs en deux dimensions. Pour le paramétrage automatique de notre bruit volumique par analyse d'un exemple, une méthode décrivant cet exemple en ellipsoïdes est nécessaire. Une décomposition en ellipsoïde d'un volume donnée, ou une méthode permettant de reconstruire les ellipsoïdes à partir de différentes prises de vue (par exemple des photographies prises selon des positions particulières connues) pourrait grandement faciliter l'édition de motifs volumiques.

Un problème similaire à celui posé par l'approche utilisée pour l'analyse en deux dimensions peut néanmoins se poser : les structures volumiques complexes peuvent nécessiter un grand nombre d'ellipsoïdes pour une reproduction adéquate. Des méthodes d'extraction d'ellipsoïdes existantes et compatibles avec notre formulation, telles que la décomposition en ondelettes évoquée précédemment, peuvent potentiellement générer un très grand nombre de gaussiennes sphériques pour reproduire des éléments pouvant être représentés par une unique gaussienne ellipsoïdale simple.

Une limite inhérente à notre approche est également que l'approximation statistique considérée par ce type d'analyse de motif est limitée à la reproduction de la forme et de l'apparence du motif à un instant fixé dans le temps. Certains des effets temporels pourraient être extraits par analyse de différents exemples séparés uniquement par un facteur de temps et être utilisés pour la définition d'un modèle adapté du bruit. L'utilisation de différentes configurations de noyaux représentants les différents états possibles des détails pourraient être utilisés pour créer cette reproduction temporelle limitée de ces effets.

L'utilisation d'une approximation statistique ne peut cependant pas reproduire correctement de nombreuses variations temporelles : prenant le cas du champ d'herbe, un grand nombre de variations locales ou globales sont liées à des phénomènes naturels précis et localisés tels que des facteurs d'humidité du sol ou d'illumination. La reproduction exacte des effets de ces phénomènes au cours du temps ne peut être que difficilement approximée efficacement par de l'aléatoire, et requiert une définition manuelle par un artiste ou une simulation physique.

Noyaux alternatifs

La littérature sur les bruits de convolution (section 1.3.3) regorge de différents noyaux choisis selon des caractéristiques ciblées. Nous avons originellement choisi un noyau construit selon une somme de gaussiennes elliptiques pour les propriétés de filtrage analytique des noyaux Gaussiens, mais les propriétés spatiales de ce noyau restent limitées. Ces limitations sont compensées pour la production de structures complexes en utilisant, dans notre cas, un grand nombre de fonctions gaussiennes, impactant fortement les performances. D'autres noyaux aux propriétés spatiales différentes permettraient cependant d'étendre l'ensemble des motifs possibles à moindre coût. Nous avons évoqué dans la conclusion du chapitre 2 une extension possible de la formulation du bruit en utilisant des courbes paramétriques comme nouvelle primitive de construction du noyau. Ces noyaux courbes permettraient, par exemple, de représenter plus efficacement des craquelures, mais également des éléments courbés complexes dans le cas des détails volumiques.

Modélisation de structures multi-échelles

L'un de nos objectifs est de rajouter à la volée des détails de surface en fonction du niveau de détails voulu dans l'image. Nous avons abordé dans cette thèse la génération de motifs structurés par une distribution contrôlée d'éléments structurels à l'apparence fixé. Dans les motifs naturels, des micro-détails peuvent normalement devenir visibles sur ces mêmes éléments lorsque l'utilisateur se rapproche de leurs surfaces, similairement aux veines des feuilles ou des brins devenant visible vues de très près. Notre modèle permet de gérer efficacement la génération d'un type de détails de surface, mais la décoration de ces détails générés procéduralement reste un problème difficile : réaliser cette décoration à l'aide d'un autre bruit procédural implique la gestion de dépendances spatiales supplémentaires. Cette décoration est d'autant plus complexe lorsque ces micro-détails sont eux même composés de différents niveaux de structures.

Une extension possible de notre modèle de bruit pour la création de structures multi-échelles serait de définir une arborescence de noyaux caractérisant les dépendances existantes entre les différents niveaux de structures. La figure A montre un premier exemple de bruit multi-échelle défini par une telle arborescence : un noyau de Gabor est utilisé comme nœud principal de cette arborescence, créant un motif de bruit de Gabor. Mais lorsque l'utilisateur agrandit le motif, un deuxième niveau de structure correspondant au fils (un noyau en forme d'étoile) apparaît sur les noyaux de Gabor distribués . Un troisième niveau de structure, défini par un noyau en forme de grille, devient également

visible sur l'image de gauche. Ce premier exemple, issu d'essais préliminaires, utilise ici les nœud pères pour contrôler la densité globale d'impulsion des noeuds fils. Cette approche permettrait potentiellement la définition de structures multi-échelles complexes et ceux en étant entièrement procédurale et évaluable par pixel en temps interactif.

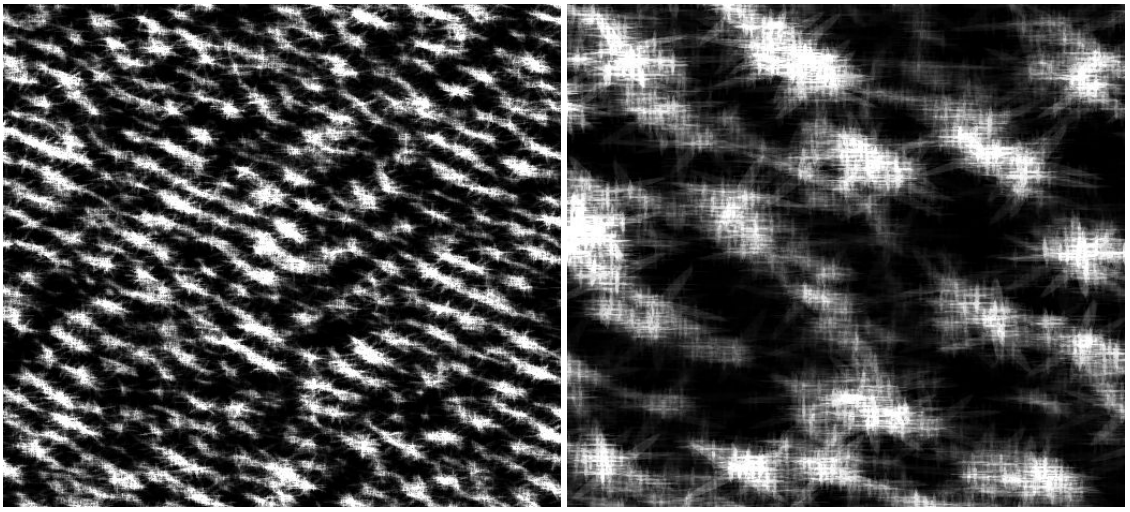


FIGURE A – Un exemple de bruit comportant différents niveaux de structures créées par une arborescence de noyau.

Bibliographie

[Ama84] AMANATIDES J.: Ray tracing with cones. *SIGGRAPH Comput. Graph. 18*, 3 (Jan. 1984), 129–135. 1.4.3

[AMH08] AKENINE-MÖLLER T., HAINES E.: *Real-Time Rendering - 3rd Edition*. A K Peters, 2008, ch. 13, pp. 575–644. 1.1.1

[AWF95] ANDREW W. FITZGIBBON R.: A buyer's guide to conic fitting. In *Proc. 5th British Machine Vision Conference, Birmingham* (1995), pp. 513–522. 2.3.2

[BCF*05] BEHRENDT S., COLDITZ C., FRANZKE O., KOPF J., DEUSSEN O.: Realistic real-time rendering of landscapes using billboard clouds. In *IN COMPUTER GRAPHICS FORUM (2005). (PROCEEDINGS OF EUROGRAPHICS* (2005). 1.1.2, 1.1.3

[BGAM12] BARRINGER R., GRIBEL C. J., AKENINE-MÖLLER T.: High-quality curve rendering using line sampled visibility. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 162:1–162:10. (document), 1.3, 1.1.1, 1.1.1

[BH02] BAKAY B., HEIDRICH W.: Real-time animated grass. In *Proceedings of Eurographics (short paper)* (2002). (document), 1.1.3, 1.9

[BHN07] BRIDSON R., HOURIHAM J., NORDENSTAM M.: Curl-noise for procedural fluid flow. In *ACM SIGGRAPH 2007 Papers* (New York, NY, USA, 2007), SIGGRAPH '07, ACM. 1.3.2, 1.3.4

[BN11] BRUNETON E., NEYRET F.: A survey of non-linear pre-filtering methods for efficient and accurate surface shading. *IEEE Transactions on Visualization and Computer Graphics* (2011). 1.1.4

[BNW*95] BOURGUET J. M., NANCY T., WEI S. J., LEROY J., CRAPPE R. G.: A variant of cooley-tuckey algorithm with local memory management. In *Proceedings of the 1995 European Conference on Design and Test* (Washington, DC, USA, 1995), EDTC '95, IEEE Computer Society, pp. 7–. 1.3.1

[BPB09] BOULANGER K., PATTANAIK S. N., BOUATOUCH K.: Rendering grass in real time with dynamic lighting. *IEEE Comput. Graph. Appl. 29*, 1 (Jan. 2009), 32–41. (document), 1.1, 1.2, 1.1.1, 1.1.2, 1.1.3, 1.1.4, 1.11, 1.1.4, 1.2.1, 1.12, 1.2.1

[BW06] BANISCH S., WÜTHRICH C. A.: Making grass and fur move. *Journal of WSCG 14*, 1-3 (2006), 25–32. 1.1.3

[CNS*11] CRASSIN C., NEYRET F., SAINZ M., GREEN S., EISEMANN E.: Interactive indirect illumination using voxel cone tracing. *Computer Graphics Forum (Proceedings of Pacific Graphics 2011) 30*, 7 (sep 2011). 1.4.3, 1.4.3

[Coo86] COOK R. L.: Stochastic sampling in computer graphics. *ACM Trans. Graph. 5*, 1 (Jan. 1986), 51–72. 1.2.1, 1.2.1

[Cra11] CRASSIN C.: *GigaVoxels: A Voxel-Based Rendering Pipeline For Efficient Exploration Of Large And Detailed Scenes*. PhD thesis, University of Grenoble, 2011. (document), 1.4.1, 1.34

[CSHD03] COHEN M. F., SHADE J., HILLER S., DEUSSEN O.: Wang tiles for image and texture generation. *ACM Trans. Graph. 22*, 3 (July 2003), 287–294. 1.2.1

[CTW*04] CHEN Y., TONG X., WANG J., LIN S., GUO B., SHUM H.-Y.: Shell texture functions. In *ACM Transactions on Graphics (TOG)* (2004), vol. 23, ACM, pp. 343–353. 1.1.3, 1.1.4

[DG97] DISCHLER J.-M., GHAZANFARPOUR D.: A procedural description of geometric textures by spectral and spatial analysis of profiles. *Computer Graphics Forum 16* (1997), C129–C139. 1.3.5

[DHI*13] DUPUY J., HEITZ E., IEHL J.-C., POULIN P., NEYRET F., OSTROMOUKHOV V.: Linear efficient antialiased displacement and reflectance mapping. *ACM Trans. Graph. 32*, 6 (Nov. 2013), 211:1–211:11. 1.2.2

[dLvL97] DE LEEUW W., VAN LIERE R.: Divide and conquer spot noise. In *Proceedings of the 1997 ACM/IEEE Conference on Supercomputing* (New York, NY, USA, 1997), SC '97, ACM, pp. 1–13. 2.2.1, 2.2.1

[DN09] DECAUDIN P., NEYRET F.: Volumetric billboards. *Computer Graphics Forum 28*, 8 (2009), 2079–2089. (document), 1.1.3, 1.7, 1.1.4

[DvGNK99] DANA K. J., VAN GINNEKEN B., NAYAR S. K., KOENDERINK J. J.: Reflectance and texture of real-world surfaces. *ACM Trans. Graph. 18*, 1 (Jan. 1999), 1–34. 1.1.2

[EHK*04] ENGEL K., HADWIGER M., KNISS J. M., LEFOHN A. E., SALAMA C. R., WEISKOPF D.: Real-time volume graphics. In *ACM SIGGRAPH 2004 Course Notes* (New York, NY, USA, 2004), SIGGRAPH '04, ACM. (document), 1.4.2, 1.32, 1.33

[Eve01] EVERITT C.: *Interactive Order-Independent Transparency*. Tech. rep., Nvidia, 2001. 1.4.2

[EWM*98] EBERT D. S., WORLEY S., MUSGRAVE F. K., PEACHEY D., PERLIN K., MUSGRAVE K. F.: *Texturing and Modeling : A Procedural Approach*, 2nd ed. Academic Press, Inc., Orlando, FL, USA, 1998. 1.2.3

[Fil09] FILIP J.: Bidirectional texture function modeling : A state of the art survey. *IEEE Transactions on Pattern Analysis and Machine intelligence 31*, 11 (november 2009). 1.1.2

[FLHS15] FAN Z., LI H., HILLESLAND K., SHENG B.: Simulation and rendering for millions of grass blades. In *Proceedings of the 19th Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2015), i3D '15, ACM, pp. 55–60. (document), 1.1.1, 1.1.1, 1.2.1, 1.2.1, 1.13

[FP07] FOREST V., PAULIN M.: Rendu temps réel de fonction de textures bidirectionnelles. *Technique et Science Informatiques 26* (2007), 975–997. 1.1.2

[GAMD10] GUETAT A., ANCEL A., MARCHESIN S., DISCHLER J.-M.: Pre-integrated volume rendering with non-linear gradient interpolation. *Visualization and Computer Graphics, IEEE Transactions on 16*, 6 (Nov 2010), 1487–1494. 3.3.1

[GD09] GILET G., DISCHLER J.: A framework for interactive hypertexture modelling. *Computer Graphics Forum 28*, 8 (2009), 2229–2243. (document), 1.2.3, 1.20

[GD10] GILET G., DISCHLER J.-M.: An image-based approach for stochastic volumetric and procedural details. In *Proceedings of the 21st Eurographics Conference on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2010), EGSR'10, Eurographics Association, pp. 1411–1419. 1.3.5

[GDG12a] GILET G., DISCHLER J.-M., GHAZANFARPOUR D.: Multi-scale assemblage for procedural texturing. *Comp. Graph. Forum 31*, 7pt1 (Sept. 2012), 2117–2126. (document), 1.2.1, 1.16, 1.2.3, 1.2.4

[GDG12b] GILET G., DISCHLER J.-M., GHAZANFARPOUR D.: Multiple kernels noise for improved procedural texturing. *Vis. Comput. 28*, 6-8 (June 2012), 679–689. (document), 1.3.3, 1.29, 1.3.4

[Gla04] GLANVILLE R. S.: Texture bombing. In *GPU Gems*. 2004, ch. 20. 1.2.1, 1.15a

[GLD*08] GOVINDARAJU N. K., LLOYD B., DOTSENKO Y., SMITH B., MANFERDELLI J.: High performance discrete fourier transforms on graphics processors. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing* (Piscataway, NJ, USA, 2008), SC '08, IEEE Press, pp. 2:1–2:12. 1.3.1

[GLLD12] GALERNE B., LAGAE A., LEFEBVRE S., DRETTAKIS G.: Gabor noise by example. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2012) 31*, 4 (July 2012). To appear. (document), 1.30, 1.3.4, 1.3.5

[GSV*14] GILET G., SAUVAGE B., VANHOEY K., DISCHLER J.-M., GHAZANFARPOUR D.: Local random-phase noise for procedural texturing. *ACM Trans. Graph. 33*, 6 (Nov. 2014), 195:1–195:11. (document), 1.3.3, 1.27, 1.3.4, 1.31, 1.3.4, 1.3.5, 2.1

[GY13] GUO X., YANG G.: Animating prairies simulation with shell method in real-time. *Journal of Software 8*, 12 (december 2013). 1.1.3

[HMDM08] HAJJAR J.-F. E., MARCHESIN S., DISCHLER J.-M., MONGENET C.: Second order pre-integrated volume rendering. In *Visualization Symposium, 2008. PacificVIS '08. IEEE Pacific* (2008), pp. 9–16. 3.3.1

[HMVI11] HENDRIKX M., MEIJER S., VELDEN J. V. D., IOSUP A.: Procedural content generation for games: A survey. In *ACM Transactions on Multimedia Computing, Communications and Applications (ACM TOMCCAP)* (July 2011). 1.2

[IYLV13] IP C. Y., YALÇIN M. A., LUEBKE D., VARSHNEY A.: Pixelpie: Maximal poisson-disk sampling with rasterization. In *Proceedings of the 5th High-Performance Graphics Conference* (New York, NY, USA, 2013), HPG '13, ACM, pp. 17–26. 1.2.1

[JBL*06] JANG Y., BOTCHEN R. P., LAUSER A., EBERT D. S., GAITHER K. P., ERTL T.: Enhancing the Interactive Visualization of Procedurally Encoded Multifield Data with Ellipsoidal Basis Functions. *Computer Graphics Forum 25*, 3 (2006), 587–596. 1.3.5, 2.2.2

[JM16] JABLONZKY S., MARTYN T.: Real-time voxel rendering algorithm based on screen space billboard voxel buffer with sparse lookup textures. In *Proceedings of WSCG* (August 2016). 1.4.3

[KB04] KOBBELT L., BOTSCH M.: A survey of point-based techniques in computer graphics. *Comput. Graph. 28*, 6 (Dec. 2004), 801–814. 1.4.2

[KB12] KUHNERT T., BRUNNETT G.: Fur shading and modification based on cone step mapping. *Comp. Graph. Forum 31*, 7pt1 (Sept. 2012), 2011–2018. (document), 1.1.1, 1.4

[KCODL06] KOPF J., COHEN-OR D., DEUSSEN O., LISCHINSKI D.: Recursive wang tiles for real-time blue noise. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2006) 25*, 3 (2006), 509–518. (document), 1.2.1, 1.14

[KCYM14] KONIARIS C., COSKER D., YANG X., MITCHELL K.: Survey of texture mapping techniques for representing and rendering volumetric mesostructure. *Journal of Computer Graphics Techniques 3*, 2 (2014), 18–60. 1.1.3

[KK89] KAJIYA J. T., KAY T. L.: Rendering fur with three dimensional textures. *SIGGRAPH Comput. Graph. 23*, 3 (July 1989), 271–280. 1.1.3

[KW03] KRUGER J., WESTERMANN R.: Acceleration techniques for gpu-based volume rendering. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)* (Washington, DC, USA, 2003), VIS '03, IEEE Computer Society, pp. 38–. 1.4.1, 1.4.3

[LD05] LAGAE A., DUTRÉ P.: A procedural object distribution function. *ACM Trans. Graph. 24*, 4 (Oct. 2005), 1442–1461. 1.2.1

[LD11] LAGAE A., DRETTAKIS G.: Filtering solid gabor noise. *ACM Trans. Graph. 30*, 4 (July 2011), 51:1–51:6. 1.3.3

[LEQ*07] LU A., EBERT D. S., QIAO W., KRAUS M., MORA B.: Volume illustration using wang cubes. *ACM Transactions on Graphics (TOG) 26*, 2 (2007), 11. 1.2.1

[Lew89] LEWIS J. P.: Algorithms for solid noise synthesis. *SIGGRAPH Comput. Graph. 23*, 3 (July 1989), 263–270. (document), 1.3.3, 1.24, 1.3.3, 1.3.5, 2.2.1

[LHN05] LEFEBVRE S., HORNUS S., NEYRET F.: Octree textures on the gpu. In *GPU Gems 2* (march 2005), Addison-Wesley. 1.1.3, 3.2.2

[LLC*10] LAGAE A., LEFEBVRE S., COOK R., DEROSE T., DRETTAKIS G., EBERT D., LEWIS J., PERLIN K., ZWICKER M.: A survey of procedural noise functions. *Computer Graphics Forum 29*, 8 (December 2010), 2579–2600. 1.2.3, 1.2.4, 1.3, 1.3.1, 1.21b, 1.22b, 1.3.5

[LLDD09] LAGAE A., LEFEBVRE S., DRETTAKIS G., DUTRÉ P.: Procedural noise using sparse gabor convolution. In *ACM SIGGRAPH 2009 papers* (New York, NY, USA, 2009), SIGGRAPH '09, ACM, pp. 54:1–54:10. (document), 1.3.3, 1.26, 1.3.4, 2.2.1, 2.4, 3.2.2

[Llo82] LLOYD S. P.: Last squares quantization in pcm. *IEEE Transactions on Information Theory 28*, 2 (march 1982). 1.2.1

[LM07] LONG J., MOULD D.: Improved image quilting. In *Proceedings of Graphics Interface 2007* (New York, NY, USA, 2007), GI '07, ACM, pp. 257–264. 1.2.1

[LPF01] LENGYEL J., PRAUN E., FINKELSTEIN A.: Real-time fur over arbitrary surfaces. In *2001 ACM Symposium on Interactive 3D Graphics* (2001), pp. 227–232. (document), 1.1.3, 1.8, 1.2.1, 1.2.1

[MB13] MCGUIRE M., BAVOIL L.: Weighted blended order-independent transparency. *Journal of Computer Graphics Techniques (JCGT) 2*, 2 (December 2013), 122–141. 1.4.4, 3.5, 3.3.3

[MCTB12] MAULE M., COMBA J. L. D., TORCHELSEN R., BASTOS R.: Transparency and anti-aliasing techniques for real-time rendering. In *Graphics, Patterns and Images Tutorials (SIBGRAPI-T), 2012 25th SIBGRAPI Conference on* (Aug 2012), pp. 50–59. 1.4.2

[MET*14] MARTIN T., ENGEL W., THIBIEROZ N., YANG J., LACROIX J.: Tressfx : Advanced real-time hair rendering. In *GPU Pro 5: Advanced Rendering Techniques*, Engel W., (Ed.). CRC Press, 2014, pp. 193–209. 1.4.2

[MN98] MEYER A., NEYRET F.: Interactive volumetric textures. In *Eurographics Rendering Workshop 1998* (New York City, NY, July 1998), Drettakis G., Max N., (Eds.), Eurographics, Springer Wein, pp. 157–168. ISBN 3-211-83213-0. 1.1.3, 1.1.3

[MNP01] MEYER A., NEYRET F., POULIN P.: Interactive rendering of trees with shading and shadows. In *Proceedings of the 12th Eurographics Conference on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2001), EGWR'01, Eurographics Association, pp. 183–196. 1.1.2

[MSR16] MAISCH S., SKANBERG R., ROPINSKI T.: A comparative study of mipmapping techniques for interactive volume visualization. *Journal of WSCG 24*, 2 (aug. 2016), 35–42. 1.1.3

[Nie13] NIESSNER M.: *Rendering Subdivision Surfaces using Hardware Tessellation*. PhD thesis, Friedrich-Alexander University of Erlangen-Nuremberg, 2013. 1.2.2

[NK03] NAGY Z., KLEIN R.: Depth-peeling for texture-based volume rendering. In *Pacific Graphics 2003* (Oct. 2003). 3.3.4

[NKB13] NEULANDER I., KATO T., BEASON K.: Rendering fur in life of pi. In *ACM SIGGRAPH 2013 Talks* (New York, NY, USA, 2013), SIGGRAPH '13, ACM. 3.4.1

[NL13] NIESSNER M., LOOP C.: Analytic displacement mapping using hardware tessellation. *ACM Trans. Graph. 32*, 3 (July 2013). 1.2.2

[OHH*02] OLANO M., HART J. C., HEIDRICH W., MARK B., PERLIN K.: Real-time Shading Languages. SIGGRAPH Courses 36, 2002. 1.3.3

[OL81] OPPENHEIM A. V., LIM J. S.: The Importance of Phase in Signals. *Proceedings of the IEEE 69*, 5 (May 1981), 529–541. 2.1

[Ola05] OLANO M.: Modified noise for evaluation on graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware* (New York, NY, USA, 2005), HWWS '05, ACM, pp. 105–110. 1.3.2

[Pan14] PANGERL D.: Screen-space grass. In *GPU Pro 5: Advanced Rendering Techniques*, Engel W., (Ed.). CRC Press, 2014, pp. 221–232. (document), 1.1.2, 1.2.3, 1.19

[Pap15] PAPAVASILIOU D.: Real-time grass (and other procedural objects) on terrain. *Journal of Computer Graphics Techniques (JCGT) 4*, 1 (February 2015), 26–49. (document), 1.1.1, 1.15b, 1.2.1, 1.17, 1.2.2

[PBFJ05] PORUMBESCU S. D., BUDGE B., FENG L., JOY K. I.: Shell maps. *ACM SIGGRAPH 2005 Papers* (2005), 626–633. 1.1.3

[PC01] PERBET F., CANI M.-P.: Animating prairies in real-time. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics* (New York, NY, USA, 2001), I3D '01, ACM, pp. 103–110. 1.1.2, 1.1.2

[PCOS10] PIETRONI N., CIGNONI P., OTADUY M., SCOPIGNO R.: Solid-texture synthesis: A survey. *IEEE Comput. Graph. Appl. 30*, 4 (July 2010), 74–89. 1.2.3

[Pel04] PELZER K.: Rendering countless blades of waving grass. In *GPU Gems* (march 2004), Addison-Wesley, pp. 107–121. (document), 1.1.2, 1.5

[Per85] PERLIN K.: An image synthesizer. *SIGGRAPH Comput. Graph. 19*, 3 (July 1985), 287–296. (document), 1.18, 1.2.3, 1.3.1, 1.3.2, 1.21, 2.1

[Per02] PERLIN K.: Improving noise. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2002), SIGGRAPH '02, ACM, pp. 681–682. 1.3.2

[PFH00] PRAUN E., FINKELSTEIN A., HOPPE H.: Lapped textures. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 465–470. 1.2.1

[PGD*16] PAVIE N., GILET G., DISCHLER J.-M., GALIN E., GHAZANFARPOUR D.: Volumetric spot noise for procedural 3d shell texture synthesis. In *Proceedings of Computer Graphics & Visual Computing* (2016). To appear. 3

[PGDG16] PAVIE N., GILET G., DISCHLER J.-M., GHAZANFARPOUR D.: Procedural texture synthesis by locally controlled spot noise. In *Proceedings of WSCG* (August 2016). 2

[PH89] PERLIN K., HOFFERT E. M.: Hypertexture. *SIGGRAPH Comput. Graph. 23*, 3 (July 1989), 253–262. 1.2.3

[PH10] PHARR M., HUMPHREYS G.: Volume scattering. In *Physically Based Rendering, Second Edition*. 2010, ch. 11, pp. 575–601. 1.4.1

[PN01] PERLIN K., NEYRET F.: Flow Noise. 28th International Conference on Computer Graphics and Interactive Techniques (Technical Sketches and Applications), Aug. 2001. 1.3.2, 1.3.4

[QCCL12] QIU H., CHEN L., CHEN J. X., LIU Y.: Dynamic simulation fo grass field swaying in wind. *Journal of Software 7*, 2 (february 2012). 1.1.1, 1.1.2, 1.1.2, 1.1.4

[RB85] REEVES W. T., BLAU R.: Approximate and probabilistic algorithms for shading and rendering structured particle systems. *SIGGRAPH Comput. Graph. 19*, 3 (July 1985), 313–322. 1.1.1

[Ree83] REEVES W. T.: Particle systems, a technique for modeling a class of fuzzy objects. *ACM Trans. Graph. 2*, 2 (Apr. 1983), 91–108. 1.2.1

[RPZ02] REN L., PFISTER H., ZWICKER M.: Object space EWA surface splatting: A hardware accelerated approach to high quality point rendering. In *Computer Graphics Forum (Eurographics 2002)* (Sept. 2002), pp. 461–470. 3.2.1

[RSEB*00] REZK-SALAMA C., ENGEL K., BAUER M., GREINER G., ERTL T.: Interactive volume on standard pc graphics hardware using multi-textures and multi-stage rasterization. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware* (New York, NY, USA, 2000), HWWS '00, ACM, pp. 109–118. 1.4.2

[RV06] RUIJTERS D., VILANOVA A.: Optimizing gpu volume rendering. In *WSCG - Winter School of Computer Graphics* (Feb 2006), vol. 14, pp. 9–16. 1.4.3

[SKB09] SPJUT J. B., KENSLER A. E., BRUNVAND E. L.: Hardware-accelerated gradient noise for graphics. In *Proceedings of the 19th ACM Great Lakes Symposium on VLSI* (New York, NY, USA, 2009), GLSVLSI '09, ACM, pp. 457–462. (document), 1.3.2, 1.22

[SKP05] SHAH M. A., KONTINNEN J., PATTANAIK S.: Real-time rendering of realistic-looking grass. In *Proceedings of the 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia* (New York, NY, USA, 2005), GRAPHITE '05, ACM, pp. 77–82. (document), 1.6, 1.1.2

[SMDS11] SALMON J. K., MORAES M. A., DROR R. O., SHAW D. E.: Parallel random numbers: As easy as 1, 2, 3. In *2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC)* (Nov 2011), pp. 1–12. 1.2.1

[SSKE05] STEGMAIER S., STRENGERT M., KLEIN T., ERTL T.: A simple and flexible volume rendering framework for graphics-hardware-based raycasting. In *Fourth International Workshop on Volume Graphics, 2005.* (June 2005), pp. 187–241. 1.4.3

[Sta98] STAM J.: Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 395–404. 1.2.2

[SV14] SALVI M., VAIDYANATHAN K.: Multi-layer alpha blending. In *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2014), I3D '14, ACM, pp. 151–158. 1.4.2

[vW91] VAN WIJK J. J.: Spot noise texture synthesis for data visualization. *SIGGRAPH Comput. Graph. 25*, 4 (July 1991), 309–318. (document), 1.3.3, 1.25, 1.3.5, 2.2.1, 2.2.1

[WAT92] WESTIN S. H., ARVO J. R., TORRANCE K. E.: Predicting reflectance functions from complex surfaces. *SIGGRAPH Comput. Graph. 26*, 2 (July 1992), 255–264. (document), III

[WE98] WESTERMANN R., ERTL T.: Efficiently using graphics hardware in volume rendering applications. In *Proceedings of the 25th Annual Conference on Com-*

puter Graphics and Interactive Techniques (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 169–177. 1.4.2

[Wes90] WESTOVER L.: Footprint evaluation for volume rendering. *SIGGRAPH Comput. Graph. 24*, 4 (Sept. 1990), 367–376. 1.4.2

[Wha05] WHATLEY D.: Toward photorealism in virtual botany. In *GPU Gems 2* (march 2005), Addison-Wesley, pp. 7–25. 1.1.2

[WLKT09] WEI L.-Y., LEFEBVRE S., KWATRA V., TURK G.: State of the art in example-based texture synthesis. In *Eurographics 2009 State of the Art Report* (2009). 1.2.3

[WM03] WELSH T., MUELLER K.: A frequency-sensitive point hierarchy for images and volumes. In *Visualization, 2003. VIS 2003. IEEE* (Oct 2003), pp. 425–432. 2.2.2, 2.5

[Wro14] WRONSKI B.: Volumetric fog : Unified compute shader vased solution to atmospheric scattering. ACM SIGGRAPH 2014 - Talk, 2014. 1.4.3

[Wym16] WYMAN C.: Stochastic layered alpha blending. In *ACM SIGGRAPH 2016 Talks* (New York, NY, USA, 2016), SIGGRAPH '16, ACM, pp. 37:1–37:2. 1.4.2

[YL08] YOON J.-C., LEE I.-K.: Stable and controllable noise. *Graphical Models 70*, 5 (2008), 105 – 115. (document), 1.28, 1.3.4

[YNBH11] YU Q., NEYRET F., BRUNETON E., HOLZSCHUCH N.: Lagrangian Texture Advection: Preserving both Spectrum and Velocity Field. *IEEE Transactions on Visualization and Computer Graphics 17*, 11 (Nov. 2011), 1612–1623. (document), 1.3.2, 1.23

[ZPvBG01] ZWICKER M., PFISTER H., VAN BAAR J., GROSS M.: Ewa volume splatting. In *Visualization, 2001. VIS '01. Proceedings* (Oct 2001), pp. 29–538. 1.4.2, 3.2, 3.2.1, 3.3.4

Si tu veux avancer dans l'étude de la sagesse, ne refuse point,
sur les choses extérieures, de passer pour imbécile et pour insensé.
Epictète