MASTER 2

INFORMATIQUE, SYNTHÈSE D'IMAGE ET CONCEPTION GRAPHIQUE

GÉNÉRATION PROCÉDURALE DE CHAMP D'HERBES

NICOLAS PAVIE

SIR - XLIM

Encadré par
GUILLAUME GILET
DJAMCHID GHAZANFARPOUR

14 Juin 2012

Remerciement

Je tiens avant tout à remercier Guillaume GILET et Djamchid GHAZANFARPOUR de m'avoir accepté, puis de m'avoir encadré et guidé tout au long du stage.

Je remercie également les membres de l'équipe SIR de XLIM, qui m'ont accompagné toutes ces années, et ont accepté de lire ce mémoire et d'évaluer mon travail.

Je tiens aussi à remercier Olivier TERRAZ d'avoir pu négocier des bureaux pour tous les stagiaires.

Un grand merci à mes amis et collègues de bureaux, Evans, Anthony et Angélique, pour leurs conseils et leur bonne humeur durant et en dehors de ce stage.

Finalement, je remercie particulièrement mes amis et ma famille qui m'ont soutenu ces derniers mois. Merci a Richard, GG et GB pour leurs discussions éclairées et les moments de détentes passés ensemble.

Résumé

Dans une scène de végétation, l'herbe est l'un des éléments le plus présent. La création manuelle de plaine d'herbe est un travail long et répétitif, et le résultat final demande beaucoup de mémoire pour être conservé. De nombreux modeleurs 3D intègrent des outils permettant de créer automatiquement de l'herbe à partir de paramètres. Mais ces outils permettent rarement de modéliser interactivement le champ d'herbes créé, et ce champ demande toujours une grande quantité de mémoire pour être stocké.

Pour répondre à ces problèmes, nous proposons une modélisation de champ d'herbes basée sur un bruit procédurale volumique. Cette modélisation est évaluable en temps constant et ne nécessite que peu de mémoire pour sa sauvegarde et sa réutilisation. Pour l'utiliser plus facilement, nous proposons une méthode de rendu volumique adaptée, permettant de créer interactivement un champ d'herbes et d'en obtenir un rendu en temps réel.

Table des matières

Table des figures

Chapitre 1

Introduction

1.1 Contexte

L'informatique graphique est un domaine encore très jeune mais extrêmement sollicité dans de nombreux domaines scientifiques pour la représentation de l'information. On observe le plus souvent les évolutions de cette branche de l'informatique à travers les films et les jeux vidéos, mais aussi pour la visualisation scientifique.

L'avancée des technologies en matière de visualisation et de création d'images de synthèse, ainsi que la croissance des puissances de calcul disponibles sur les ordinateurs récents, se sont accompagnés d'un besoin grandissant de réalisme, que ce soit pour les jeux vidéos, les films ou l'ensemble des domaines utilisant l'informatique graphique.

L'industrie de la création graphique a évolué en conséquence, apportant aux artistes des outils et logiciels de modélisations de plus en plus performants. Ces outils intègrent de nos jours de nombreuses fonctions, permettant par exemple d'affiner un maillage automatiquement, ou de générer une partie des objets à partir de paramètres définis par l'utilisateur.

Cependant, une grande partie du travail de modélisation est encore réalisé manuellement, et le résultat reste coûteux en taille mémoire, même en utilisant les outils de modélisation actuels. Ces problèmes apparaissent en particulier pour une scène contenant énormément de détails visuels, tel que les scènes de végétation comportant de grandes plaines d'herbes.

1.2 Problématiques

Lors de la réalisation d'un scène, le niveau de réalisme de celle-ci dépend du niveau de détails des objets, ce qui correspond à la structure visible de l'objet. La représentation d'un objet est couramment décris par trois niveaux de structures : les macrostructures, les millistructures et les microstructures. Gilet [8] ajoute à cette classification les mésostructures qui sont un niveau de détails intermédiaire entre les macrostructures et les millistructures, permettant d'enrichir visuellement un objet (figure 1.1) .

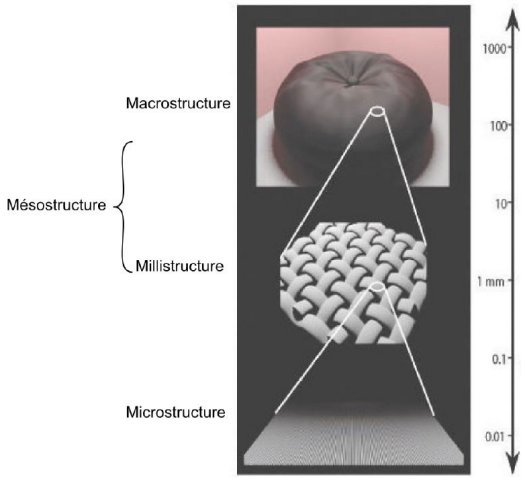


FIGURE 1.1 – Niveaux de détails structurelles d'un objet [8]

Nous nous intéressons plus particulièrement aux détails liés aux millistructures et aux mésostructures. La création de ces détails posent plusieurs problèmes de conception :

- La complexité de la réalisation : Les infographistes et autres créateurs doivent généralement créer "manuellement" les détails d'une scène ou des objets qu'elle comporte. Même pour les plus expérimentés d'entre eux, la création de tous ces détails est une tâche difficile, ce travail demandant une grande minutie. Cette complexité s'accompagne naturellement d'une augmentation du temps de réalisation de la scène : la création d'un élément de détail d'une scène peut être un travail long à réaliser, qu'il faut souvent multiplier par la grande quantité de détails demandée.

- Le stockage mémoire : Cette problématique touche tous les utilisateurs et créateurs de scènes 3D. L'ajout de détails augmente en conséquence la quantité d'informations liées à la scène. Les textures et les modèles requièrent alors un plus grand espace de stockage en mémoire.

- Un affichage non-interactif : Enfin, dernière problématique liée à ces détails, la limite de mémoire et la complexité des scènes empêchent souvent leur utilisation dans un milieu interactif, tel que les jeux vidéos. La quantité et la qualité des détails de la scène peuvent être trop important, ou requièrent des schémas complexes d'optimisation pour que les cartes graphiques puissent les gérer dans leurs totalités en conservant une bonne fluidité.

Il existe cependant des outils permettant de répondre à ces problématiques : les *méthodes procédurales*. Ces méthodes permettent de modéliser un ensemble d'objets à partir d'une fonction dont les paramètres déterminent certaines propriétés du résultat final.

De nombreux logiciels de modélisation 3D intègrent des fonctions de *modélisation procédurale*, tel que les L-system, qui permettent de créer un objet ou un ensemble d'objet à partir de paramètres où de scripts. Ces fonctions peuvent simplifier la réalisation de différents champs d'herbe ou de la fourrure en 3D, en générant un modèle géométrique à partir des paramètres définis par l'utilisateur.

Mais ces fonctions sont itératives et complexes à calculer, ce qui les rend non évaluables rapidement ou en temps constant. Elles ne peuvent être utilisées que pour précalculer un modèle géométrique qui doit être stocké en mémoire.

Avec l'évolution de la puissance de calcul disponible dans les ordinateurs actuels, il est possible d'étendre les méthodes de modélisation procédurale à un autre type de méthode : les méthodes d'*évaluation procédurale*. Ces méthodes d'évaluation nécessitent une certaine puissance de calcul pour être utilisable de façon interactive, mais un matériel graphique récent permet d'obtenir un rendu temps-réel fluide.

Ces méthodes ne génèrent plus un modèle géométrique à priori, mais per-

mettent son évaluation en chaque point de l'espace. Le modèle géométrique n'a plus besoin d'être stocké, il suffit de conserver la fonction et ses paramètres, puis de l'appliquer sur un volume ou une surface. Ceci permet entre autre de ne pas avoir à regénérer un modèle géométrique à chaque modification. L'utilisation de méthodes d'évaluation procédurale représente donc une économie conséquente de temps et de stockage.

La plupart de ces méthodes d'évaluation sont déjà utilisées en 2D, afin par exemple de créer (ou reproduire) interactivement des textures à partir de bruits. Mais nous allons montrer qu'il est possible d'utiliser ces méthodes procédurales pour créer en temps réel des objets 3D, et plus particulièrement des détails de mésostructures tel que de l'herbe ou les feuillages dans une scène.

1.3 Description du sujet

Le but de ce stage est d'accroitre le réalisme de scènes naturelles par ajout procédural de détails en 3D. Ces détails permettent d'ajouter un réalisme complémentaire à certains éléments et scènes complexes, tels que l'écorces des arbres ou les plaines de végétation.

Des travaux traitent de la génération de détails en 2D en se basant sur des fonctions pseudo-aléatoires dites "de bruit", tel que le bruit de Gabor introduit par Lagae et al. [17]. Ce bruit apporte l'avantage de pouvoir être paramétré, mais ses paramètres sont peu intuitifs. Pour pallier ce problème, il existe des méthodes d'acquisition (à partir de photographie ou de dessin [11] [9]) pour faciliter ce paramétrage, mais la définition de paramètres d'un phénomène 3D complexe reste un problème difficile.

L'objectif du projet est de générer au vol, grâce a un bruit procédural, de grandes surfaces de végétation, en se basant dans un premier temps sur la génération d'un champ d'herbes.

1.4 Contribution

A travers ce mémoire, nous présentons une nouvelle méthode de représentation d'un champ d'herbes, généré automatiquement à partir de paramètres. Nos contributions se situent à la fois au niveau de la modélisation et au niveau du rendu de celle-ci :

- Une modélisation procédurale d'un champ d'herbes Nous présentons tout d'abord dans ce mémoire une nouvelle approche de la modélisation d'un champ d'herbes, évaluable en n'importe quel point de l'espace et paramétrable. Ce modèle repose sur les caractéristiques des bruits procéduraux, ainsi que sur la possibilité de décomposition de ces bruits en noyaux, introduite par les *sparse convolution noises*. Cette modélisation permet alors l'utilisation disjointe de différents modèles de brins et de dispersions, chacun de ces modèles pouvant offrir eux aussi une grande variété de résultats grâce à leurs paramètres.

- Une méthode de rendu volumique adaptée Le rendu volumique est très complexe à réaliser, en particulier pour des modèles 3D procéduraux qui sont continus dans l'espace. Les méthodes tels que le *ray marching* ne proposent qu'une approximation de ces modèles. Nous présentons une méthode de rendu adaptée à notre modélisation, basée sur une recherche d'échantillons significatifs dans un rayon de vue, nous permettant une représentation précise du modèle procédurale à moindre coût.

1.5 Organisation du document

Nous commençons par présenter dans la première partie les travaux et outils qui nous ont permis de concevoir la modélisation. Cette partie fait état des différentes modélisations de champ d'herbes existantes les plus utilisées, introduit plus en détail le concept de méthode procédurale, et présente les bruits procéduraux existants.

Nous présentons ensuite notre contribution propre, en commençant par expliciter les outils utilisés, avant d'appréhender la nouvelle modélisation et la méthode de rendu utilisée.

Nous détaillons notre modélisation en présentant l'effet des différents paramètres sur le résultat. La modélisation ensuite est testée en utilisant une méthode de *ray marching* et notre méthode de rendu optimisée. Pour mesurer le gain de performance apporté par notre méthode, nous comparons un résultat obtenu par *ray marching* à celui obtenu par notre méthode de rendu avec les mêmes paramètres.

Chapitre 2

Etat de l'art

Notre objectif est de modéliser un champ d'herbes qui soit évaluable en n'importe quel point de l'espace et pouvant être paramétré. Il existe déjà de nombreuses modélisations de champ d'herbes, associées à certaines représentations des données, mais ces modélisations requièrent pour chacune d'entre elle de stocker en mémoire une quantité d'informations pouvant être conséquente selon la taille de la surface et le type de représentation.

Nous voulons que notre champ d'herbes soit généré de manière automatique à partir de paramètres. Notre modélisation s'inspire donc des méthodes procédurales de modélisation et d'évaluation existantes.

Il est important pour nous d'obtenir un champ d'herbes qui paraisse réaliste, c'est à dire sans percevoir d'effets de répétitions. Les outils tel que les bruits procéduraux permettent de répondre à ce type de problématique.

Nous commençons par passer en revue différentes modélisations et représentations d'un champ d'herbes dans la section 2.1, triées selon le type de représentation utilisé. La section 2.2 présente le concept de méthode procédurale, argumenté de quelques exemples afin de mieux en cerner les grands principes. Enfin, la section 2.3 récapitule les bruits procéduraux les plus utilisés.

2.1 Représentations et modélisations de champ d'herbes

Dans une scène de végétation, l'herbe est l'élément se trouvant souvent en plus grand nombre, avec des milliers voir des millions de brins de tailles et de formes différentes. Il existe de nombreuses représentations possibles d'un brin

ou d'un ensemble de brins, associées à des méthodes de modélisations. Boulanger [3] dénombre principalement trois familles de représentation : les représentations surfaciques, les représentations basées image, et les représentations volumiques.

Il est courant d'utiliser différentes représentations simultanément selon le niveau de détails voulu. Boulanger [3] utilise deux familles précédemment cités pour représenter un champ d'herbe : une représentation surfacique pour les brins proches, et une représentation volumique pour les brins à moyenne et longue portées.

2.1.1 Représentations surfaciques

Les représentations surfaciques utilisent des primitives géométriques simples, tels que des triangles ou des quadrilatères, pour décrire la surface d'un objet. Ce type de représentation permet d'obtenir un rendu réaliste d'un objet et d'y associer des textures et une méthode d'illumination, ce qui en fait une représentation très utilisée pour des brins proches de la caméra.

De nombreux logiciels de modélisation existent de nos jours et permettent de créer manuellement chaque brin d'herbe, ou un groupe de brin. La création brin par brin d'une scène demande alors un temps considérable, et l'utilisation d'un groupe de brins répété sur la surface laisse clairement apparaitre l'effet de recopie. En définitif, ce type de modélisation n'est vraiment utile que pour les cas de brins très spéciaux. De plus, la plupart des logiciels de modélisation permettent de nos jours d'utiliser la génération procédurale de champ d'herbe.

Pour la modélisation procédurale de brins d'herbe, Reeves et Blau [31] propose d'utiliser un système de particule : un brin d'herbe est décomposé en points générés à partir de la simulation du mouvement d'une particule au cours du temps (figure 2.1a). Ces points donnent alors la forme du brin. Il est aussi possible d'utiliser une courbe de Hermite pour déterminer les vecteurs représentatifs du brins [1] (figure 2.1b).

Avec leur système de particules, Reeves et Blau proposent de relier les coordonnées des particules par des lignes anti-aliasées pour reproduire le brin. Une autre représentation proposée par Boulanger [3] consiste à utiliser une bande de quadrilatères texturés semi-transparents, en remplacement des lignes anti-

aliasées (figure 2.2).

Une représentation utilisant la courbe de Hermite consiste à former une chaine de billboards colorés. Le brin d'herbe est découpé en plusieurs billboards sur sa longueur, et la position et l'orientation de chaque billboard sont évalués grâce à la formule de la courbe de Hermite [12] (figure 2.1c).

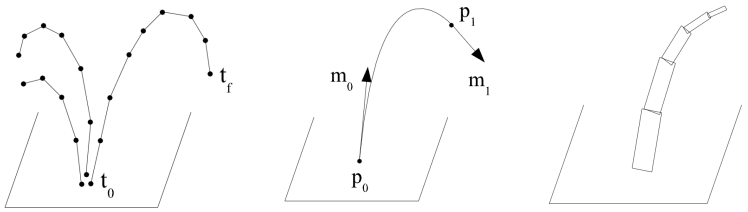


FIGURE 2.1 – Modélisations géométriques des brins.[3]
(a) : Système de particule. (b) : Courbe de Hermite. (c) : Chaine de billboards

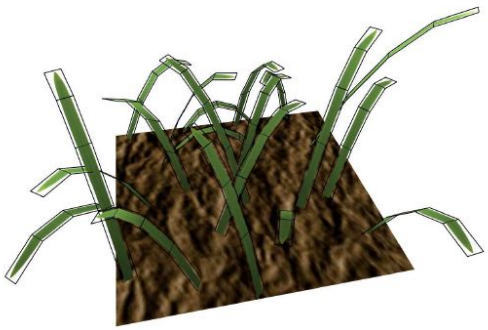


FIGURE 2.2 – Représentation géométrique par Boulanger [3]

Dans l'industrie du cinéma, une représentation commune des brins est celle d'un objet standard : chaque brin d'herbe est composé d'un ensemble de faces triangulaires. Mais cette méthode est inappropriée pour le rendu temps-réel, la quantité de triangles générés augmentant rapidement avec le nombre de brins et pouvant vite dépasser les capacités du matériel utilisé.

Conclusion

L'utilisation d'une géométrie pour représenter les brins d'herbes permet d'obtenir un rendu très précis, en plus de facilement pouvoir faire varier les modèles de brins.

Mais elle présente des inconvénients très gênant pour le rendu temps-réel : dans le cas d'une caméra éloignée du champ d'herbe, le nombre de primitives à traiter peut devenir ingérable pour la carte graphique. Cette complexité est inutile pour des brins inférieurs à la taille d'un pixel, ces primitives n'étant pas discernables les une des autres dans ce cas et, et génère des effets d'aliassage.

Ce type de représentation, de par le cout en mémoire et le temps de calcul qu'il engendre, n'est souvent utilisé que pour un rendu proche de l'observateur dans le cas du rendu temps-réel.

2.1.2 Représentations basées image

Les représentations basées image utilisent des textures semi-transparentes pour représenter un objet ou un ensemble d'objet. Ces textures sont appliquées sur des polygones simples pour en faire le rendu. Le rendu d'un simple billboard avec une texture semi-transparente faisant toujours face à la caméra est depuis longtemps utilisé dans le jeux vidéo [33]. Contrairement aux représentations surfaciques, ce type de représentation n'a pas pour but l'exactitude visuelle du résultat, mais une économie de mémoire et de calcul.

Pour le rendu d'herbes très lointaines, la méthode la plus couramment utilisée est l'utilisation d'une ou plusieurs textures 2D plaquées sur le sol. Cependant cette représentation ne permet pas de créer des effets d'animation ou d'illumination réalistes.

La représentation de brins par un ou plusieurs billboards texturés est très utilisée dans les jeux, son faible coût en mémoire et en calcul étant un atout pour le rendu temps-réel. Le rendu d'un brin ne correspond alors qu'au rendu de quelques primitives texturées. L'utilisation d'une texture semi-transparente permet de représenter des objets non-rectangulaires, et avec la capacité de calcul par pixel disponible, il est possible d'utiliser des modèles d'illumination de plus en plus précis. Cependant, le mouvement de la caméra laisse clairement

apparaitre l'absence de l'effet de parallaxe.

Pour récupérer cet effet, Pelzer [22] utilise plusieurs billboards entrecroisés (figure 2.3). Il dispose aléatoirement un grand nombre de ces structures aléatoirement sur le sol pour éviter l'effet de répétition, caché par le grand nombre d'intersections entre les billboards.

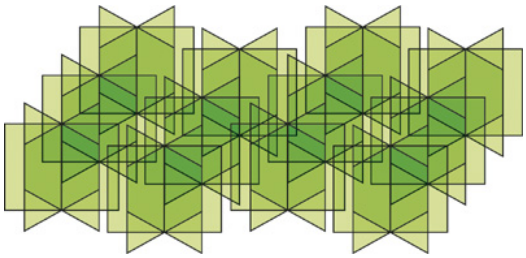


FIGURE 2.3 – Représentation d'herbes par billboards entrecroisés. [22]

Une représentation plus simple consiste à utiliser des couches de billboards alignés (figure 2.4). Des couches orthogonales sont définies pour camoufler l'apparence de la structure lors du mouvement de la caméra [23]. Cette méthode laisse cependant apparaitre de nombreux artefacts, en particulier pour une vue de dessus de l'herbe où la structure utilisée est clairement visible.

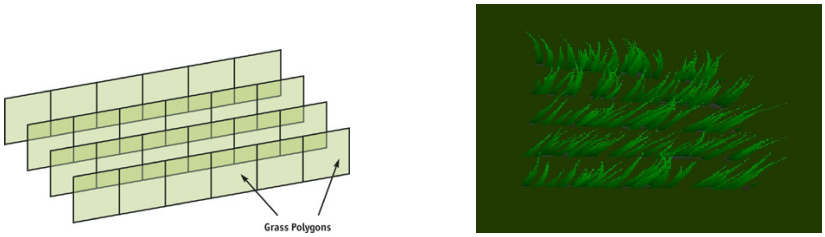


FIGURE 2.4 – Représentation d'herbes par billboards alignés. [22] [23]

Conclusion

Les représentations basées sur des images sont surtout efficaces pour un rendu à moyenne et longue distance de la caméra.

Le rendu du sol en plaquant dessus une ou plusieurs textures est très utilisé pour le rendu d'herbes trop lointaines pour être distinguées normalement. Mais il pose de nombreux problèmes pour représenter l'illumination et l'animation du champ d'herbes de manière réaliste, et l'utilisation d'une BTF pour conserver ces propriétés serait un procédé trop coûteux en mémoire.

La représentation par billboards texturés permet un rendu très rapide à courte et moyenne portées, mais manquant de réalisme et montrant des répétitions de brins identiques. Certaines représentations peuvent faire apparaitre de nombreux artefacts lors du rendu, alors que d'autres laisseront clairement apparaître la structure utilisée et négligeront l'effet de parallaxe. Ces défauts sont particulièrement visibles pour un petit nombre de brins affichés.

2.1.3 Représentations volumiques

Les représentation volumiques ne décrivent pas simplement la surface d'un objet, mais aussi sa structure interne. Cette structure est souvent recréer par découpe de l'objet pour en stocker les différentes couches. Ce type de représentation permet d'afficher des objets complexes contenant beaucoup de détails, et cela en conservant un effet de parallaxe complet contrairement aux représentations basées image. Cependant, ces représentations nécessitent l'élaboration de méthodes de rendu adéquates.

De la même manière que pour le rendu de fourrure par Lengyel et al. [19], il est possible d'utiliser une série de sur-couches texturées semi-transparentes pour donner l'illusion d'un rendu volumique. Bakay et al. [2] utilise aussi des sur-couches mais ils n'utilisent qu'une seule texture pour le rendu du sol et des brins d'herbes de différentes hauteurs. La méthode de rendu associée à cette représentation consiste à rendre couche par couche le volume ainsi créé (figure 2.5).

A l'instar de Boulanger [3], il est possible d'utiliser un ensemble de coupe pour la représentation volumique de l'herbe. La modélisation par coupes consiste à découper le volume en tranches successives, lesquelles sont rendues séparément et stockées sous forme de textures ou de BTFs. Boulanger génère pour cela une successions de tranches verticales selon deux axes perpendiculaires, ainsi qu'une tranche horizontale proche du sol. La méthode de rendu consiste alors, de la même manière que pour les billboards, à afficher ces tranches textu-

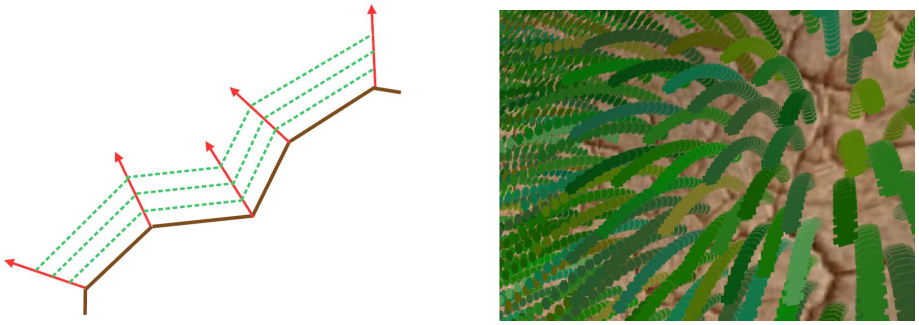


FIGURE 2.5 – Représentation par sur-couches texturées.[2]

rées semi-transparentes. Pour le rendu de l'herbe lointaine, seule la tranche horizontale est utilisée pour le rendu, ce qui permet d'améliorer les performances (figure 2.6).

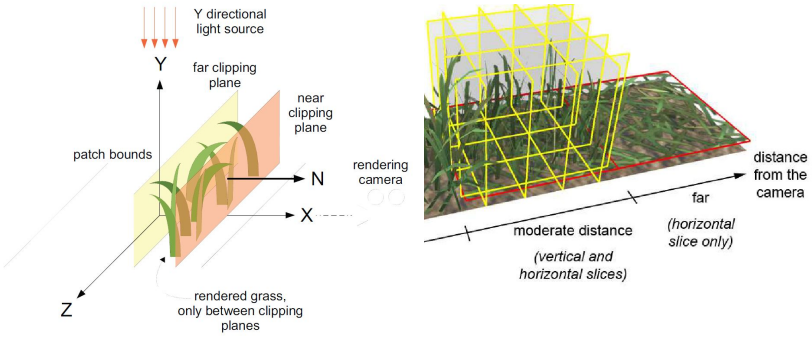


FIGURE 2.6 – Représentation volumique par la méthode de Boulanger. [3]

Une autre représentation possible est l'utilisation de ce que Neyret et Decaudin [5] appelle un billboard volumique. Pour le créer, un ensemble d'affichages en coupe de l'objet sont réalisés selon 6 directions de vues différentes, afin de créer une grille de voxels. Une fois le billboard créé, Neyret et Decaudin utilisent comme support des prismes triangulaires possédant des coordonnées de textures 3D. Le rendu de ces primitives est réalisé également par un rendu par coupe du volume le long du rayon de vue (figure 2.7).

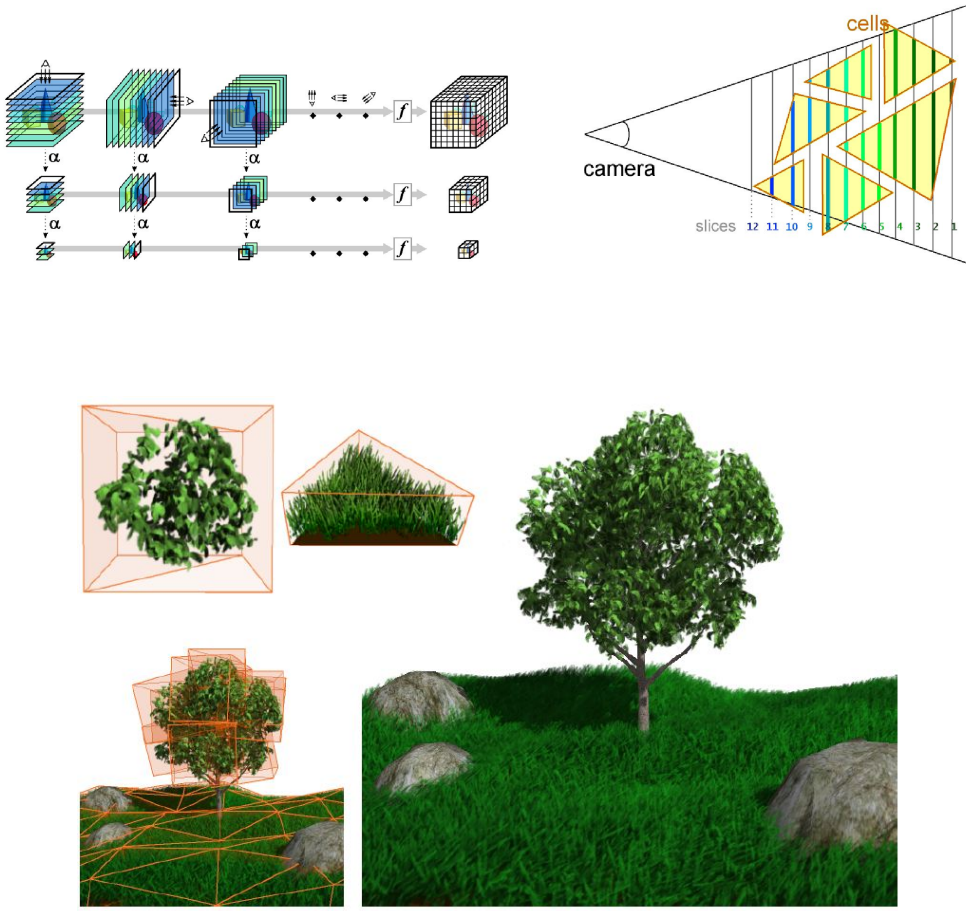


FIGURE 2.7 – Représentation et rendu par billboards volumiques. [5]

Conclusion

Dans le cas de l'utilisation de sur-couches texturées, la rastérisation complète de tous les quadrilatères composant les sur-couches demande une importante quantité de calcul. Les représentations par tranches texturées et par billboards volumiques quand à elles peuvent demander très grande quantité de mémoire selon la résolution des textures générées.

Les représentations volumiques peuvent donner des résultats très probant à moyenne de distance, puisqu'elles permettent de conserver l'effet de parallaxe contrairement aux représentations basées sur des images. Mais selon la méthode de rendu et la représentation utilisée, la puissance de calcul et la quantité de mémoire requises peuvent être conséquentes.

2.1.4 Récapitulatif

Pour réaliser le comparatif de ces différentes représentations, nous prenons en compte principalement quatre paramètres :

- La reconstitution de l'effet de parallaxe,
- La qualité visuelle globale du rendu,
- Les performances obtenues en utilisant la représentation,
- Le coût en mémoire de la représentation.

Le récapitulatif de la gestion de ces propriétés par les représentations précédemment décrites est décrit dans la table 2.1.

2.2 Les méthodes procédurales

Pour faciliter la création de scènes complexes, de nombreux logiciels de modélisation proposent des outils spéciaux afin de créer de la fourrure ou de l'herbe facilement. Ces outils sont basés sur ce que l'on appelle des méthodes procédurales : ce n'est plus l'artiste, mais l'ordinateur qui crée la scène à partir d'une "procédure" dont les paramètres sont définis par l'artiste.

Plus généralement, Perret [28] reprend comme définition qu'une méthode procédurale peut être assimilée à une amplification de ses données d'entrée, programmes ou paramètres . Cette distinction entre programmes et paramètres

Type de rendu	Représentation	Parallaxe	Qualité visuelle	Performances	Coût mémoire	Utilisation privilégiée
Géométries	Lignes anti-aliasées	Oui	0	-	-	courtes distances
	Triangles	Oui	++	-	–	
	Chaine de billboards	Oui	+	-	-	
	Bande de quadrilatères	Oui	+	-	-	
Images	Texture plaquée	Non	–	++	++	longue et moyenne distance
	Billboard simple	Non	-	+	+	
	Billboards croisés	Oui*		+	+	
	Billboards alignés	Oui*	-	++	+	
Volumes	Sur-couches texturées	Oui**	+	-	–	moyenne distance
	Slicing alignés sur les axes	Oui**	+	+	-	
	Billboards volumiques	Oui**	++	++	—	

Le symbole + (respectivement -) indique une gestion favorable (respectivement défavorable) du paramètre par la méthode.
* : Le parallaxe n'est géré que par approximation si plusieurs couches de billboards sont utilisés.
** : La gestion de l'effet de parallaxe est dépendant de la résolution des données.

TABLE 2.1 – Récapitulatif des différentes représentations d'herbes.

d'entrée permet de prendre en compte les méthodes utilisant des scripts de modélisation.

Cette définition englobe de nombreuses méthodes liées à la génération procédurale de contenu dans une scène, mais toutes ne sont pas destinées à la même utilisation. Pour la génération procédurale de contenu, nous distinguons deux cas d'utilisation : la génération hors-rendu, et la génération pendant le rendu. La génération hors-rendu signifie que le contenu est entièrement modélisé avant d'être sauvegardé ou utilisé. Les méthodes de modélisation vu dans la section précédente sont de cet ordre. La génération pendant le rendu implique que le contenu est "évalué" en même temps que l'affichage.

De la même manière, nous distinguons donc les méthodes de génération procédurales en deux catégories :

- **les méthodes de modélisation procédurales** qui sont utilisées pour générer un élément avant d'en faire le rendu : la méthode génère l'élément qui sera gardé en mémoire et sera réutilisable pour le rendu.
- **les méthodes d'évaluation procédurales** qui sont utilisées pour générer l'élément au moment du rendu : la méthode est évaluée pendant l'affichage et renvoie un résultat utilisable pour le rendu.

2.2.1 La modélisation procédurale

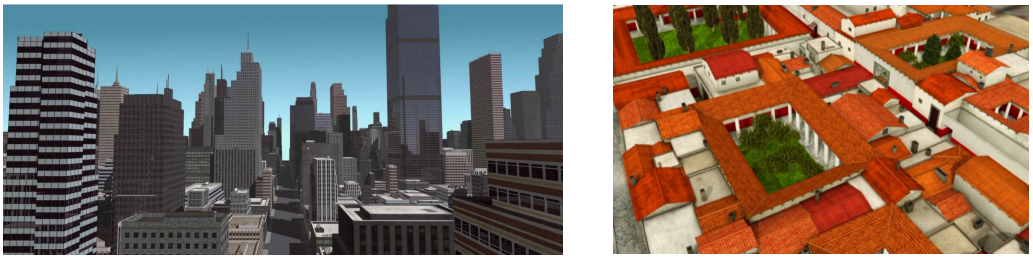
Les méthodes de modélisation procédurale sont actuellement les plus utilisées dans l'industrie du jeux vidéo et du cinéma pour la création de contenu [13] : ces méthodes facilitent la création des scènes complexes, comportant de nombreux éléments similaires.

Pour la génération de villes, les méthodes de modélisations telle que les grammaires génératives sont très souvent utilisées, certains modeleurs se spécialisant dans ce type de modélisation. Le logiciel CityEngine par exemple utilise un *L-système* étendu pour créer un ou plusieurs bâtiment à partir d'un volume englobant initial [21], tous les bâtiments appartenant au même style architectural (figure 2.8a).

Plus récemment, Müller et al. [20] ont mis au point une grammaire de formes architecturales appelée *CGA shape*, permettant de décrire architecturalement un bâtiment (figure 2.8b).

Un autre aspect particulièrement étudié est la modélisation procédurale de végétation. Pour la génération d'arbres et de plantes, Prusinkiewicz et Lindenmayer [30] détaillent quelques exemples de conceptions d'arbres à partir de L-systèmes.

En utilisant des 2Gmap L-systèmes, Peyrat et al. [29] définissent une méthode de modélisation procédurale de feuilles réalistes. Leur méthode permet de simuler la croissance et l'évolution d'une feuille au cours du temps, ainsi que les dégradations liées à l'environnement telles que les trous causés par des



(a) (b)

FIGURE 2.8 – Modélisations procédurales de villes
(a) : Manhattan (CityEngine [21]) ; (b) : Pompeii (CGA Shape [20]).

insectes (figure 2.9). Cette méthode à été étendue au 3Gmap par Terraz et al. [32] pour la modélisation procédurale de bois (figure 2.10).

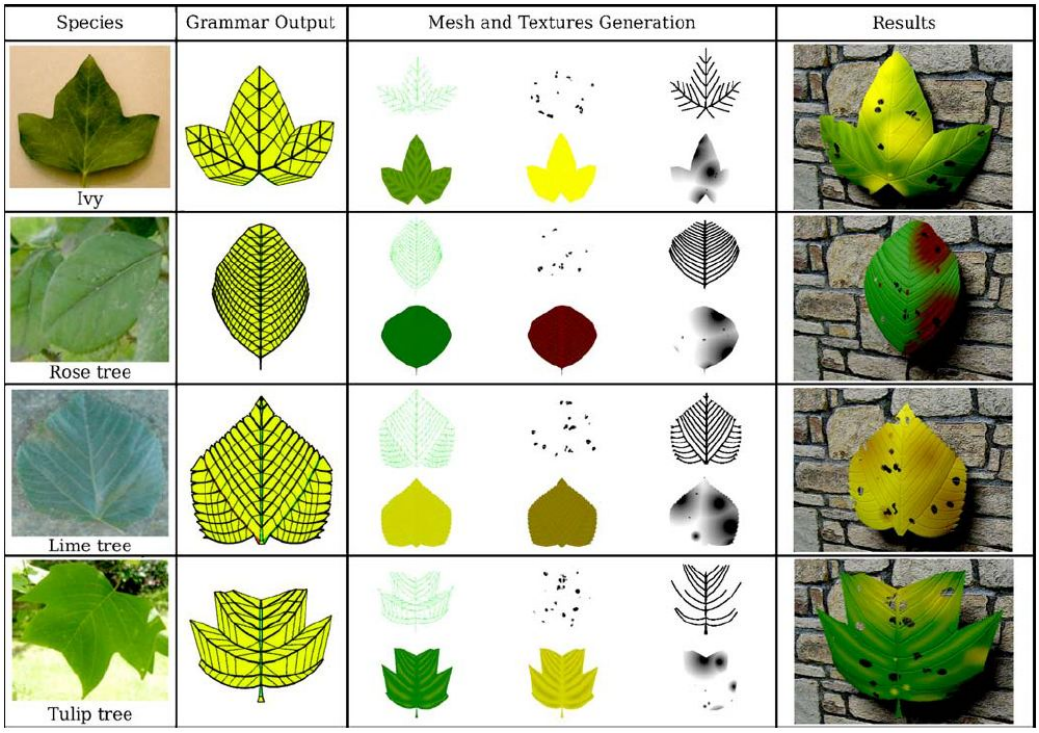


FIGURE 2.9 – Exemples de modélisation procédurale de feuilles. [29]

Les plantes constituées de répétitions du même motif à plusieurs échelles,

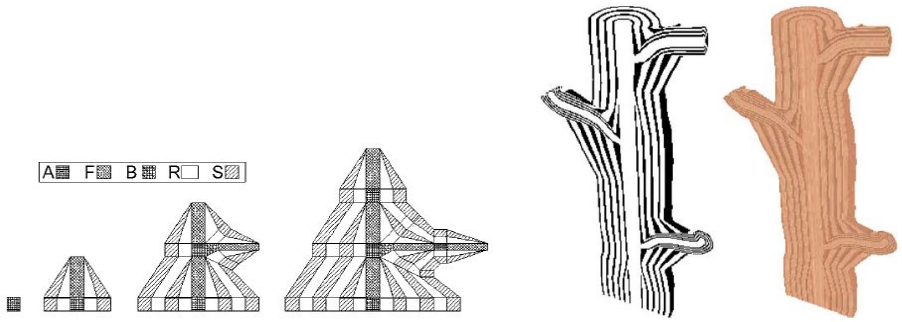


FIGURE 2.10 – Exemple de modélisation procédurale de bois. [32]

telles que les fougères, peuvent aussi être modelées de façon très précise à partir de méthodes de générations de fractales.

Conclusion

Les méthodes de modélisation procédurales sont très largement utilisées dans l'industrie et peuvent s'adapter à de nombreux domaines. La possibilité de pouvoir créer rapidement un ou plusieurs objets, avec parfois un plus grand réalisme que le résultat obtenu par un artiste, est un atout majeur de ce type de méthode.

Mais elles restent des méthodes itératives, et donc du domaine de la génération hors-rendu. Elles sont le plus souvent utilisées comme des méthodes de génération de géométries et de textures, pour créer le contenu d'une scène qui sera enregistrée. Ces méthodes peuvent nécessiter une très grande quantité de mémoire pour sauvegarder et réutiliser ce contenu, en particulier dans le cas des scènes de végétation.

2.2.2 L'évaluation procédurale

Si elles sont moins utilisées dans l'industrie des films ou des jeux vidéos pour la modélisation géométrique, les fonctions d'évaluations procédurales ont tout de même une grande utilité pour la représentation et la génération d'effets visuels variés, ainsi que pour la génération de textures.

L'évaluation procédurale permet de réaliser des effets météorologiques complexes, tel que des nuages réalistes, et ce en conservant une uniformité des temps de calculs. Neyret et al. [4] proposent dans ce domaine une méthode procédurale de rendu de nuage volumique. Leur méthode évalue le transport de la lumière en chaque point d'un maillage englobant le nuage. Grâce à une hypertexture procédurale, ils approximent la diffusion de la lumière à partir d'un point de sortie de la lumière dans le nuage (figure 2.11).

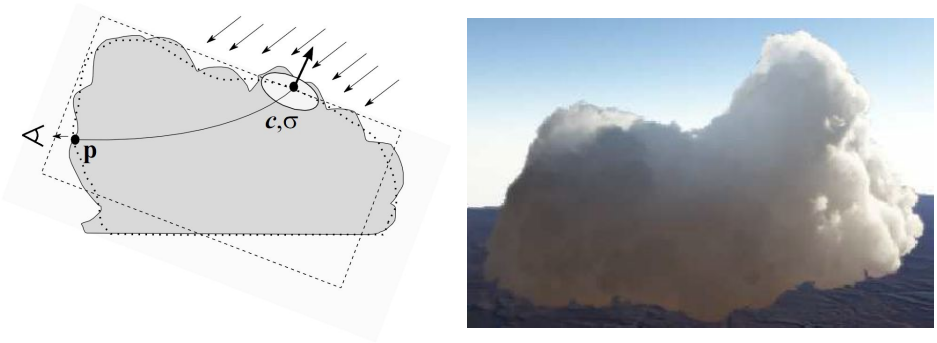


FIGURE 2.11 – Nuage procédural. [4]

Il est également possible de représenter du feu par évaluation procédurale. La méthode décrite par Fuller et al. [6] permet à un artiste de créer et de visualiser interactivement une flamme dynamique : à partir d'un motif et de paramètres définis par un artiste, leur méthode crée une flamme qui est évaluée en chaque point d'une cage modifiable par l'utilisateur. Les micro-variations et l'animation de la flamme sont ensuite générées grâce à un bruit procédural paramétrable (figure 2.12).

Conclusion

Contrairement aux méthodes de modélisation procédurales, ces méthodes sont évaluées à chaque passe de rendu. Cette caractéristique les rend particulièrement adaptées à la modélisation interactive, pour visualiser les modifications apportées par chaque paramètres.

L'avantage de ces méthodes est leur évaluation possible en n'importe quel point de l'espace et en temps constant. Il n'est pas nécessaire de stocker le ré-

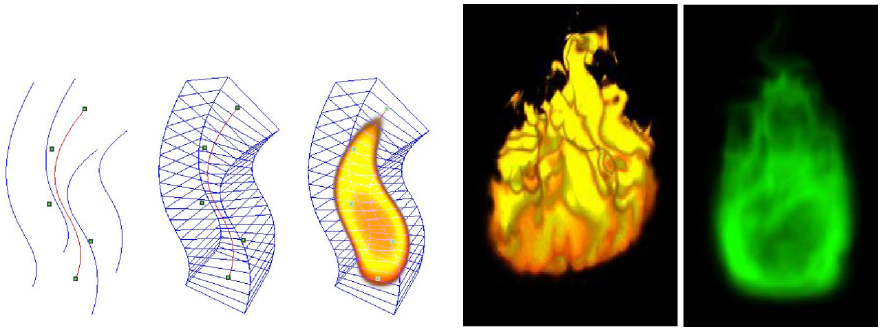


FIGURE 2.12 – Feu procédural volumique. [6]

sultat, leur coût mémoire est donc négligeable, et le rendu est plus rapidement réalisé.

2.3 Les bruits procéduraux

En informatique graphique, un bruit est une fonction aléatoire stationnaire, assimilé à un signal. Il a pour objectif de générer un motif non structuré et aléatoire. Les bruits sont particulièrement utilisés pour rajouter des détails. Ils sont par exemple très utilisés pour créer des perturbations à l'apparence naturelle dans certains effets visuels.

Les motifs aléatoires sont souvent caractérisés par des propriétés particulières :

- Dans le domaine spatial, ils sont définis par des valeurs réparties sur l'ensemble de l'espace.
- Dans le domaine fréquentiel, ils sont définis par les phases et amplitudes de toutes leurs fréquences caractéristiques.

Lagae et al. [16] estiment cependant que ces informations sont insuffisantes pour caractériser précisément un bruit. Ils favorisent la définition d'un bruit par son spectre de puissance, qui représente la magnitude des fréquences couvertes par ce bruit.

Un bruit peut être utilisé à travers plusieurs méthodes :

- Par génération d'une ou plusieurs textures stockées en mémoire. Ce type de bruit est appelé *bruit explicite*,
- Par évaluation directe d'une fonction en un point. Ce sont les *bruits procéduraux*.

Les bruits procéduraux permettent d'associer plusieurs caractéristiques des méthodes procédurales aux bruits. Mais il existe parmi ces bruits procéduraux plusieurs méthodes de génération de bruit. Lagae et al. [16] décrivent particulièrement deux familles de bruits procéduraux : les *lattice gradient noises* et les *sparse convolution noises*.

2.3.1 Caractéristiques procédurales

Un bruit procédural est donc une méthode procédurale pour simuler et évaluer un bruit. Le motif du bruit est contrôlé directement par les paramètres de la méthode.

La définition procédurale d'un bruit apporte de nombreux avantages potentiels [16] :

- Ils ne requièrent qu'un coup minime en mémoire : il n'y aucune texture à stocker contrairement aux bruits explicites.
- Ils sont continus, et ne souffrent donc pas des discontinuités liées au changement de résolution.
- Ils sont non périodiques, ce qui leur permet de recouvrir un grand espace sans laisser apparaitre de répétition.
- Ils sont paramétrables, ce qui leur permet de générer différents motifs à partir d'une même fonction.
- Ils sont évaluables en temps constant, et cela quelle que soit la position du point d'évaluation ou les évaluations précédentes.

2.3.2 Les *lattice gradient noises*

Ce type de bruit est généré par interpolation ou par convolution de valeurs aléatoires et/ou de gradients définis aux points d'une grille de valeurs entières. Le bruit de le plus connu de cette famille est le *bruit de Perlin* [24] [25] qui est depuis longtemps utilisé dans l'industrie, mais de nombreux bruits utilisant

différents types de grilles ou méthodes de calcul existent. Nous nous intéressons essentiellement au bruit de Perlin et à différentes évolutions de celui-ci, les autres bruits étant présentés plus en détails par Lagae et al. [16].

Le *bruit de Perlin* [24] [25] détermine une valeur de bruit en un point en calculant un gradient pseudo-aléatoire pour chacun des huit vecteurs les plus proches sur une grille cubique d'entiers. Ce bruit à la fois simple et rapide est l'un des plus utilisés dans les applications graphiques depuis sa création mais il est cependant fortement sujet à des effets d'aliassage ou d'artefacts directionnels dans ses premières versions.

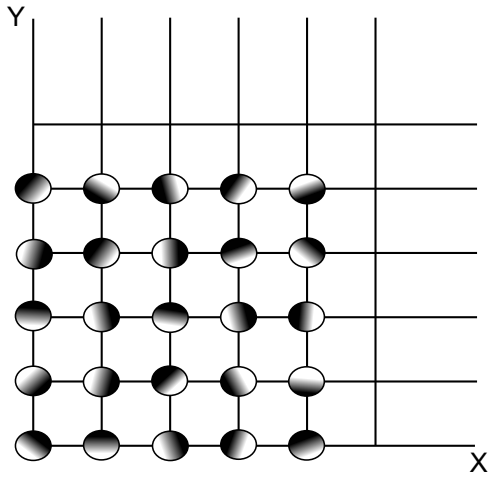


FIGURE 2.13 – Champ de gradient

Depuis sa création, ce bruit a subit des modifications majeures, dont ont découlées de nouveaux bruits. Le *simplex noise* [27] par exemple est une évolution du bruit de Perlin qui utilise une division de l'espace en simplexes pour en réduire la complexité. Plus récemment, le *better gradient noise*, introduit par Kensler et al. [14] améliore la décorrélation axiale, la limite de bandes de fréquences, ainsi que la qualité de la projection sur des surfaces 2D du bruit de Perlin.

Le bruit de Perlin s'est aussi vu adapté à différentes utilisations spécifiques tel que la simulation de fluides. Le *flow noise* mis au point par Perlin et Neyret

[26] permet par exemple de représenter des flux variants avec le temps, tel que l'écoulement de la lave.

Conclusion

La simplicité d'utilisation et la rapidité de calcul des *lattice gradient noises* en ont fait un outil majeur de l'informatique graphique. Ils apparaissent dans de nombreux domaines de représentation, et un large choix d'implémentations sur GPU est disponible.

Cependant, l'utilisation de grilles de gradients ou de valeurs limite les possibilités de définition et de contrôle du spectre de puissance du bruit. Une possibilité de contrôle existe à travers la pondération de la grille utilisée, mais cette pondération ne permet que faiblement de modifier le motif généré. Le nombre de motifs possibles en reste alors fortement limité.

2.3.3 Les *sparse convolution noises*

Cette famille de bruits se base sur la génération d'un bruit à partir d'une somme de noyaux pondérés et positionnés aléatoirement sur une surface. La définition initiale d'un *sparse convolution noise* par Lewis est la convolution d'un noyau k quelconque avec une distribution de Poisson γ éparse [16] :

$$N(x,y) = \int\int \gamma(u,v)k(x-u,y-v)\,du\,dv$$

Ce type de bruit se compose donc de deux fonctions distinctes : une fonction d'évaluation qui est le noyau du bruit, et une fonction de distribution/pondération des noyaux, souvent assimilée à un bruit blanc. Une définition plus simple peut être extraite de la formule précédente :

$$N(x,y) = k(x,y) * p(x,y) = \sum_i k(x-x_i, y-y_i)$$

Où $*$ dénote l'opérateur de convolution, $p(x,y)$ représente la fonction de distribution, et $k(x,y)$ représente la fonction d'évaluation du noyau.

Pour optimiser ce calcul du bruit, il est possible de limiter la portée de la somme aux noyaux proches du point d'évaluation. On peut pour cela utiliser une grille virtuelle, dont les cellules seraient de la taille du rayon des noyaux.

Seules la cellule contenant le point d'évaluation et les cellules voisines sont alors prises en compte.

Lagae et al. [16] précisent que la distribution de Poisson à un spectre de puissance constant. Sans modifier la fonction de distribution utilisée, le motif final du bruit est dépendant du spectre du noyau. Il existe des noyaux notables pour le contrôle qu'ils apportent au bruit, tel que le noyau de Gabor[17], permettant de généré un grand nombre de motifs.

Afin de paramétrer ces bruits, de nombreux travaux se sont attachés à reproduire un motif à partir de l'analyse spectrale d'un échantillon, soit en créant un ou plusieurs noyaux, soit en reparamétrant un noyau existant. [11] [10] [7]

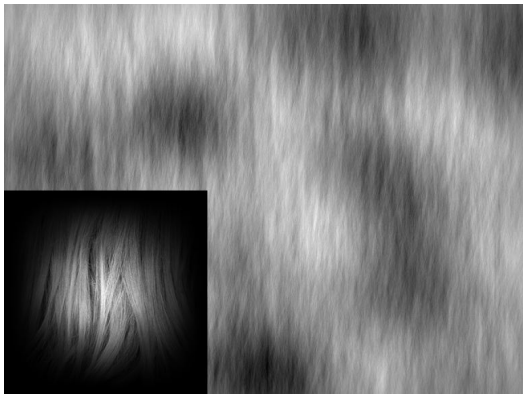
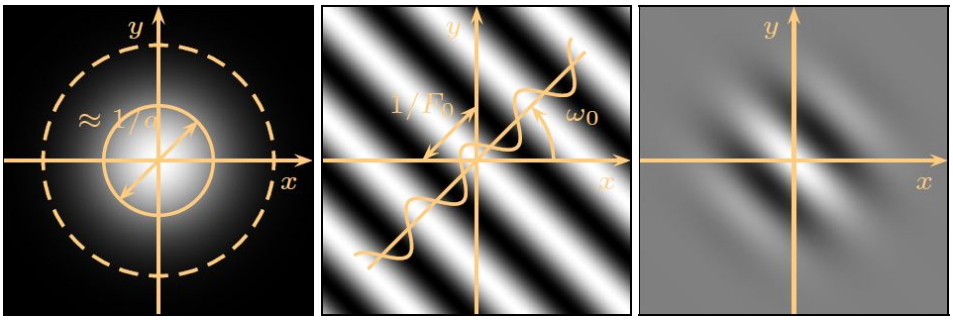


FIGURE 2.14 – Reproduction d'un motif de cheveux. [16]
Le noyau est créer par analyse spectrale d'un échantillon.

Se basant sur les *sparse convolution noises*, Lagae et al. présentent à travers une série d'articles ([17], [15], [18]) le bruit de Gabor. Le noyau de Gabor utilisé par ce bruit est originellement définis par la multiplication d'une enveloppe gaussienne et d'une harmonique (figure 2.15) :

$$g(x,y) = Ke^{-\pi a^2(x^2+y^2)}\cos[2\pi F_0(x\cos\omega_0 + y\sin\omega_0)]$$

Où K et a sont la magnitude et la largeur inverse de la gaussienne, et F_0 et ω_0 sont la fréquence et l'orientation du cosinus. Le noyau de Gabor présente l'avantage de pouvoir finement contrôler le spectre du bruit à partir de ses paramètres. Ce bruit procédural permet à la fois d'obtenir un bruit isotrope et un bruit anisotrope, le bruit isotrope étant obtenu par l'orientation aléatoire de



(a) Enveloppe gaussienne (b) Harmonique (c) Noyau de Gabor

FIGURE 2.15 – Construction du noyau de Gabor. [17]

l'harmonique du noyau. Ce bruit permet aussi d'utiliser des méthodes de filtrage pour améliorer la qualité visuel

Le paramétrage du noyau de Gabor permet de couvrir de nombreux spectres de bruits, mais la définition des paramètres afin d'obtenir un certain motif peut être fastidieuse. Dans cette optique, Gilet et al. [11] ont proposé une approche de création automatique de textures bruitées, en utilisant des exemples et le bruit de Gabor. Leur méthode se base sur une décomposition spectrale d'un motif, afin d'obtenir des paramètres adéquats pour configurer un ensemble de noyaux de Gabor.

Le bruit de Gabor et les autres *sparse convolution noises* évoqués par Lagae et al. [16] utilisent généralement des noyaux définis en 2D pour être utilisé sur des surfaces. Les travaux les plus récents sur la reproduction de motifs [11] [10] [7] créent eux aussi des noyaux définis en 2D. Nous verrons cependant qu'il est possible d'étendre la définition d'un noyau à la 3D, pour une utilisation volumique de ce type de bruit par exemple.

Conclusion

Contrairement aux *lattice gradient noises*, les *sparse convolution noises* sont non périodiques, et ne nécessitent qu'une faible quantité de mémoire pour le stockage. Mais ces bruits apportent un autre avantage dans notre cas : la séparation entre le motif des noyaux et leur positionnement permet de plus finement contrôler l'apparence du motif final. Même si ces bruits sont moins rapi-

dement évaluables, il devient possible de représenter un très grand nombre de motifs en utilisant différents noyaux et différentes distributions.

Le bruit de Gabor nous montre spécifiquement qu'il est possible d'obtenir à partir d'un seul noyau une grande variété de bruit en faisant varier ses paramètres. Ce noyau particulier apporte en plus la possibilité de filtrer le bruit pour améliorer sa qualité visuelle.

2.3.4 Récapitulatif

Plus de détails sur les différents *sparse convolution noises* sont référencés par Lagae et al. dans [16]. Nous rappelons cependant les différentes propriétés que nous prenons en compte pour notre bruit en décrivant comment chacun des bruits gèrent ces propriétés.

	Perlin [25]	Better gradient [14]	Sparse convolution [16]	Gabor [17]
Catégorie	LGN	LGN	SCN	SCN
Contrôle	Poids	Poids	Noyau	Noyau (paramètres)
Filtrage	-	-	*	+
Quantité motifs	-	-	++	+
Performance	++	+	-	-

le symbole + (respectivement -) indique une gestion favorable (respectivement défavorable) du paramètre.
* : Dépendant du noyau choisi.

TABLE 2.2 – Récapitulatif des différents bruits étudiés

Les *sparse convolution noises* ont un intérêt certain pour le contrôle possible sur leur spectre de puissance. Les travaux de Lagae et al. [17], [15], [18] montre qu'avec un seul noyau paramétrable, il est possible d'obtenir de nombreux motifs de bruit de haute qualité. De nombreux travaux portent sur la création de noyaux ou la définition de paramètres pour contrôler avec précision le motif pouvant être obtenu avec ces bruits [11] [10] [7].

Cependant, aucun des travaux précédemment cités ne définit de noyaux 3D pour la création d'un *sparse convolution noise*. Il est possible pourtant d'étendre le noyau de Gabor à une troisième dimension. Nous verrons qu'il est aussi pos-

sible de définir un nouveau noyau volumique permettant de mieux contrôler l'apparence finale d'un bruit volumique.

Chapitre 3

Méthodologie

Notre objectif est de modéliser un champ d'herbe pouvant être généré de façon automatique, qui soit réaliste et facilement paramétrable. Ce champ doit sembler réaliste à différentes échelles de visualisation, c'est à dire à une courte comme à une longue distance de vue. La solution la plus rapide, pour obtenir ce réalisme à plusieurs niveaux de détails, est de définir en premier la modélisation pour une courte distance de vue. Ce niveau de détail implique de répondre à plusieurs problématiques :

- La modélisation précise d'un brin d'herbe,
- La distribution des brins pour représenter le champ.

La représentation d'un champ d'herbes peut se décomposer en deux niveaux : l'apparence globale du champ, et les variations naturelles. L'apparence globale du champ peut être représentée grâce à des paramètres simples tels que la hauteur de ses brins ou leur couleur. Mais tous les brins d'herbes n'ont pas la même taille ni la même couleur, bien qu'ils soient très similaires. Ces variations sont liées à des phénomènes physiques complexes difficilement contrôlables, mais l'influence de ces phénomènes peut être simulée par un processus stochastique. Il nous faut donc une méthode pouvant générer un champ à partir de ses propriétés générales, tout en reproduisant les variations naturelles qu'il pourrait subir. Cette méthode, paramétrable et semi-aléatoire, correspond à la définition d'un *bruit procédural* 2.3 .

Pour répondre aux différentes problématiques posées, ce bruit procédural doit aussi nous permettre de contrôler d'un coté la modélisation des brins, et de l'autre leurs positions dans l'espace. Il nous faut donc utiliser un bruit procédural pouvant être décomposé en deux sous-fonctions : une fonction d'évaluation (ou *noyau*), et une fonction de distribution. Ce type de bruit "composé"

correspond à la définition des *sparse convolution noises* (section 2.3.3) .

La problématique de modélisation du brin requiert de trouver une fonction d'évaluation nous donnant un résultat assez réaliste, tout en nous permettant de facilement modifier l'apparence du brin. Une autre aspect que nous devons prendre en compte est la possibilité de filtrer cette fonction, afin d'améliorer la qualité visuelle selon la résolution de l'image. La formule du noyau gaussien répondant à ces conditions, nous pouvons baser notre fonction d'évaluation sur celui-ci.

3.1 Outils utilisés

Pour réaliser notre modélisation, nous utilisons plusieurs outils décris plus précisément dans la partie 2.

3.1.1 Fonction de bruit procédural

Comme dit précédemment, notre modélisation se base sur un bruit procédural (section 2.3). En nous basant sur ce type de bruit, notre modélisation hérite de différentes propriétés :

- Nous n'avons pas à stocker un volume de données important pour créer le champ d'herbe ;
- La définition du champ ne souffre donc pas de discontinuités dépendantes de la résolution de l'affichage ou de la distance de vue ;
- Le champ s'applique facilement sur l'intégralité de la surface ou du volume, en gardant une distribution réaliste ;
- Il est facile de modifier l'apparence du champ en faisant varier des paramètres ;
- Notre modélisation de champ est plus facilement traitée sur GPU ou sur processeur multi-cœur, car chaque point peut être parallèlement évalué de manière indépendante ;

3.1.2 Sparse convolution noise

Notre modélisation prend son origine dans la formule simplifiée du *sparse convolution noise* (section 2.3.3) :

$$N(x,y) = k(x,y) * p(x,y) = \sum_i k(x - x_i, y - y_i)$$

La forme particulière de l'équation de ce bruit nous permet de modéliser séparément la fonction du noyau ($k(x,y)$) de la fonction de la distribution ($p(x,y)$), les deux fonctions étant indépendantes l'une de l'autre.

Cette séparation rajoute à notre modèle une plus grande flexibilité quant à la conception de différents noyaux, nous permettant ainsi de créer plusieurs modèles de brins sans devoir modifier la fonction de distribution. Cette souplesse s'applique également à l'élaboration de plusieurs modèles de distribution des brins.

3.1.3 Le noyau gaussien

Tout comme le noyau de Gabor (figure 2.15) , la fonction d'évaluation que nous utilisons est issue d'un noyau gaussien. La forme spatiale de ce noyau est une ellipse, dont la formule initiale est :

$$G(x,y) = Ke^{-\pi a^2(x^2+y^2)}$$

Où K et a sont respectivement la magnitude et la largeur inverse de la gaussienne.

Nous verrons qu'il est cependant très simple d'adapter cette formule à un environnement 3D, ainsi que d'ajouter des paramètres de contrôle de la forme du noyau. De plus, la possibilité de filtrage de ce noyau pourra nous permettre d'obtenir un anti-aliassage naturelle lors de l'évaluation.

3.2 Modélisation du champ d'herbe

Si les différents outils ci-dessus sont définis sur deux dimensions, notre objectif est d'obtenir une représentation en trois dimensions d'un champ d'herbe,

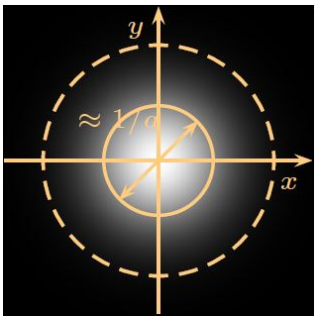


FIGURE 3.1 – Noyau gaussien. [17]

dont les brins sont distribués le long d'une surface. Nous cherchons alors à obtenir une fonction de bruit de la forme :

$$N(x,y,z) = \sum_i k(x - x_i, y - y_i, z)$$

Où x_i et y_i représentent des positions aléatoires, calculées grâce à la fonction de distribution.

Nous commencerons donc par créer un nouveau noyau pour représenter un brin d'herbe.

3.2.1 Nouveau noyau gaussien

A partir du noyau gaussien précédemment défini, il est possible de créer un nouveau noyau ellipsoïde, évalué en trois dimensions, qui soit assez paramétrable pour en contrôler la forme. Nous ajoutons pour cela des paramètres de hauteur, de largeur et d'épaisseur au noyau. Ceci nous donne la fonction :

$$k(x,y,z,l,h,w) = Ke^{-\pi a^2(\frac{x^2}{l^2}+\frac{y^2}{w^2}+\frac{z^2}{h^2})}$$

Où l, w, et h représentent respectivement la largeur, l'épaisseur et la hauteur de de l'ellipsoïde.

Ce nouveau noyau nous sert de support pour la représentation des brins d'herbes. Lors de la conception de la modélisation, nous sommes initialement partis de l'hypothèse qu'un brin d'herbe est un ellipsoïde très fin, voir plat, très allongé sur un de ses axes, et non torsadé. Ce noyau va nous permettre de reproduire et de contrôler ces différentes caractéristiques.

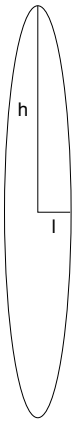


FIGURE 3.2 – Configuration du noyau

3.2.2 Distribution des brins

Afin de tester notre modélisation en condition réelle, nous utilisons une fonction de bruit blanc comme fonction de distribution. Il est néanmoins possible d'utiliser des distributions plus complexes, basées sur d'autres bruits en deux ou trois dimensions, voir même liées au temps.

La fonction de distribution a pour but de positionner les noyaux dans l'espace d'évaluation. La position de chaque noyau est actuellement calculée grâce à un générateur de nombre pseudo-aléatoire issu des travaux de Lagae et al.[17].

3.3 Méthode de rendu

Notre nouvelle modélisation étant basé sur un bruit 3D, son rendu nécessite une méthode de rendu volumique pour être le plus précis possible. La première étape nécessaire est de définir l'équation de rendu de notre modèle avant de chercher plus en avant des méthodes d'optimisations de son évaluation.

3.3.1 Équation du rendu

Nous avons une modélisation d'un champ d'herbe, représentée par une fonction de bruit procédural évaluable en tout point de l'espace :

$$N(x,y,z) = \sum_i k(x-x_i, y-y_i, z, l, h, w)$$

Pour évaluer la contribution totale de ce bruit à l'image, la fonction le représentant doit être évaluée sur la totalité de chaque rayon de vue partant des pixels de l'image. L'équation de chaque rayon est alors :

$$R(s_{in}, s_{out}) = \int_{s_{in}}^{s_{out}} N(s)ds$$

Où s_{in} est le point d'entrée du rayon dans le volume de définition du bruit, et s_{out} son point de sortie. La variable s représente un point de l'espace situé le long du rayon de vue. Le calcul d'une intégrale est complexe, mais il peut être approximer par une somme d'échantillons du rayon :

$$R(s_{in}, s_{out}) \approx \sum_{n=0}^{N_e} N(s(n))$$

Où N_e représente le nombre d'échantillon du rayon, et $s(n)$ le point correspondant à l'échantillon n du rayon. Ce point est obtenu par la formule :

$$s(n) = s_{in} + n\frac{||s_{out} - s_{in}||}{N_e}$$

L'équation finale du rayon que nous obtenons est alors :

$$R(s_{in}, s_{out}) \approx \sum_{n=0}^{N_e} \sum_{i} k(s(n)_x - x_i, s(n)_y - y_i, s(n)_z, l, h, w)$$

Cette équation donne une bonne approximation de la contribution du bruit à un rayon de vue, à condition d'utiliser un nombre d'échantillons assez important. Cette méthode d'évaluation du rayon pose aussi des problèmes de performances liés au nombre d'échantillons s'il est trop important. Cependant, des problèmes de précision apparaissent pour une quantité faible de ceux-ci, un noyau pouvant se trouver entre deux échantillons et donc ne pas être pris en compte.

Sans changer le pas d'échantillonnage du rayon, un autre problème posé par cette équation est le nombre de noyaux ne contribuant pas aux échantillons du rayon. La contribution d'un noyau étant minime voir nulle à partir d'une certaine distance d'évaluation, une optimisation possible est de limiter le nombre de noyaux à évaluer selon une distance maximale.

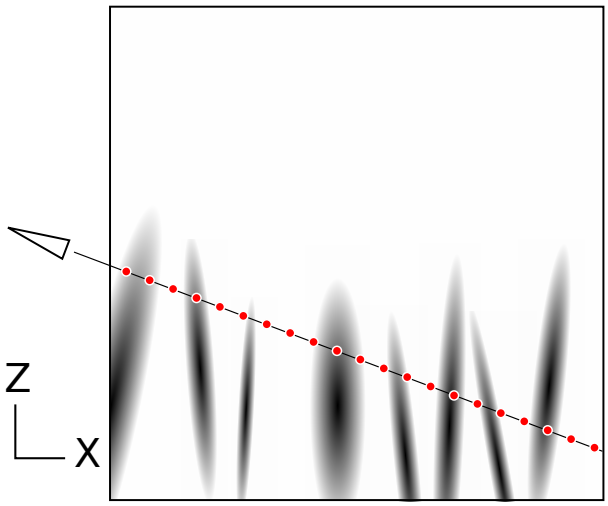


FIGURE 3.3 – Évaluation du rayon

3.3.2 Optimisation de la distribution

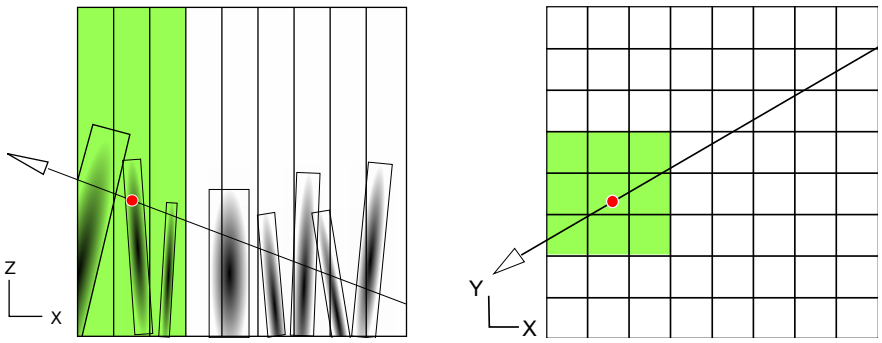
Lors de l'évaluation d'un point, nous ne désirons pas calculer la convolution de l'ensemble des noyaux en ce point, mais uniquement ceux se trouvant à portée.

Lagae et al.[16] proposent, comme méthode d'optimisation, de diviser l'espace de distribution des noyaux en cellules. La distribution des brins n'est plus faite sur l'ensemble de l'espace traversé par le rayon, mais sur la cellule contenant l'échantillon évalué et ses cellules voisines (figure 3.4).

Afin d'éviter les artefacts lors de l'évaluation, il peut être nécessaire d'adapter la taille des cellules selon la taille maximale des brins. Si la taille d'un des brins est supérieure à la taille d'une cellule, il pourrait ne pas être évalué dans les cellules voisines dans le cas où il serait couché.

Avec la division en cellule de l'espace, le parcours du rayon peut être converti en parcours de cellules. Ce parcours est alors calculé grâce à un algorithme de bresenham (figure 3.5).

Cette méthode ne règle cependant pas entièrement le problème du nombre d'échantillons évalués. Dans une même cellule, de nombreux noyaux peuvent ne pas contribuer à l'échantillon courant, même s'ils sont à portée. Une autre



La zone verte représente les cellules dans lesquelles la distribution est réalisée et servants à l'évaluation de l'échantillon rouge.

FIGURE 3.4 – subdivision de l'espace de distribution

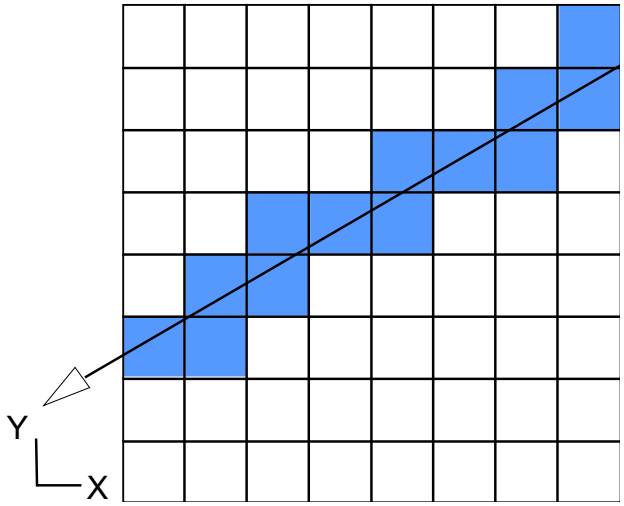


FIGURE 3.5 – Parcours de la grille

solution consiste à rechercher les échantillons significatifs du rayon pour chaque noyau de la cellule.

3.3.3 Optimisation de l'échantillonnage

Grâce aux propriétés du noyau utilisé, il est possible d'optimiser l'échantillonnage du rayon, en cherchant spécifiquement des échantillons du rayon

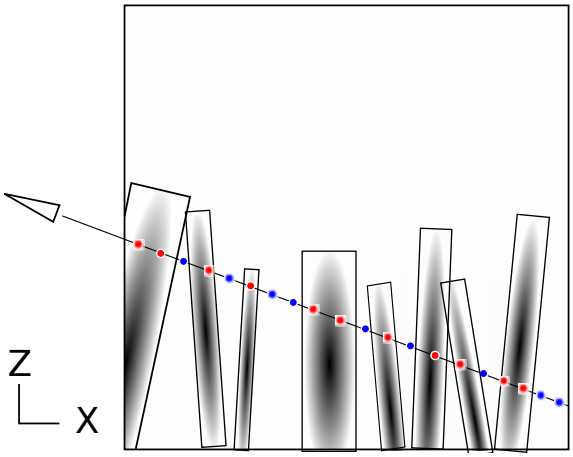


FIGURE 3.6 – Echantillonnage d'une cellule

selon les noyaux distribués dans une cellule.

Un brin d'herbe est un élément de végétation très fin, souvent représenté par un ou plusieurs objets plats (partie 2.1). Si nous considérons de même le noyau comme extrêmement fin sur l'un de ses axes, alors le noyau peut être considéré comme appartenant presque totalement à un plan. A partir de cela, il nous est facile de déterminer un point d'intersection entre le rayon et le plan sustentatoire du noyau. L'échantillon ainsi obtenu est celui ayant le plus de chance d'obtenir la contribution du noyau et de participer au résultat final. Cela nous donne l'équation du rayon :

$$R(s_{in}, s_{out}) \approx \sum_i k(s_{ix} - x_i, s_{iy} - y_i, s_{iz}, l, h, w)$$

Où le point s_i est le point d'intersection entre le plan du noyau i et le rayon de vue.

Cette recherche d'échantillon permet de réduire considérablement le nombre de calculs d'évaluation du bruit sur le rayon, ce qui permet une augmentation très nette des performances.

Cette nouvelle optimisation apporte cependant un nouveau problème. Lors de l'évaluation, tous les échantillons sont trouvés dans un ordre aléatoires dans chacune des cellules. Il n'est plus possible de distinguer l'échantillon le plus

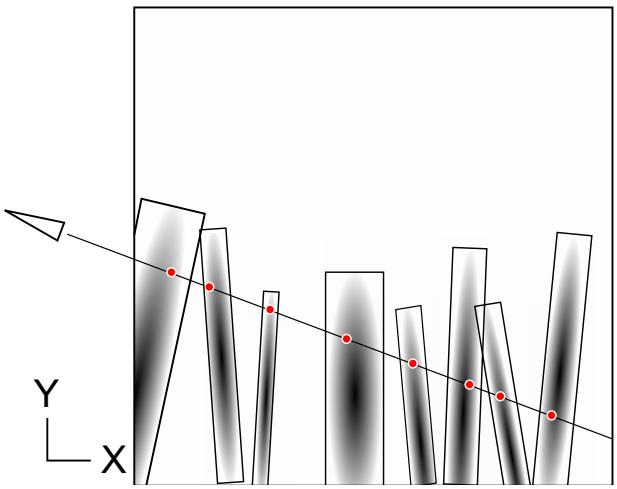


FIGURE 3.7 – Optimisation de l'échantillonnage

proche du plus éloigné.

3.3.4 Simulation de la profondeur

Notre seconde optimisation pose un problème de visibilité. Les échantillons sont sommés sans tester leur profondeur, il est donc impossible de distinguer le brin le plus proche du plus éloignés. Ce mélange des échantillons enlève l'effet de parallaxe, et de ce fait le réalisme du champ généré.

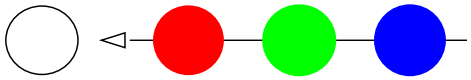
Nous utilisons une distribution des brins aléatoires, les échantillons obtenus par notre méthode de recherche ne sont donc pas évalués dans l'ordre de leur profondeur le long du rayon. Pour simuler la profondeur dans ce cas, nous utilisons l'*Order Independent Transparency*. Cette méthode consiste à utiliser une heuristique pour donner aux brins proches plus de chances d'apparaitre que les brins éloignés.

Nous calculons cette heuristique en utilisant l'information de profondeur de chaque l'échantillon. Cette heuristique est ensuite appliqué comme une pondération supplémentaire, qui influencera la contribution de l'échantillon au résultat final. L'équation du rayon en utilisant l'*OIT* est alors :

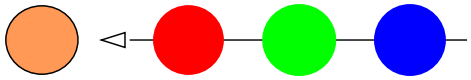
$$R(s_{in}, s_{out}) \approx \frac{\sum_i w_{oit}(s_i) k(s_{ix} - x_i, s_{iy} - y_i, s_{iz}, l, h, w)}{\sum_i w_{oit}(s_i)}$$

Où w_{oit} représente l'heuristique du point s_i. Un exemple de calcul de cet heuristique d'un point s sur le rayon pourrait être :

$$w_{oit}(s) = \frac{1}{||s - s_{in}|| + 1}$$



(a) sans OIT



(b) avec OIT

FIGURE 3.8 – Simulation de profondeur

Chapitre 4

Resultats

Nous présentons dans cette partie les résultats obtenus avec notre modélisation. Pour illustrer son fonctionnement, nous commençons par expliciter l'effet de chacun des paramètres du noyau sur le champ créé. Nous précisons ensuite les variations naturelles simulées par la méthode, qui permettent d'obtenir un champ à l'apparence réaliste.

Nous présentons ensuite les résultats de notre modélisation à travers deux méthodes de rendu : une méthode de *ray marching* et notre méthode de rendu optimisée. Notre modélisation a été initialement conçu en utilisant la méthode de *ray marching*, mais cette méthode pose des problèmes de performances pour la modélisation interactive. Nous présentons alors notre méthode de rendu optimisé, permettant de rendre un nombre plus important de brins de manière fluide.

La modélisation et les différentes méthodes de rendu ont été implémentées sur GPU, puis testées sur un ordinateur équipé d'une carte graphique GTX 580 et d'un processeur i7 2600k.

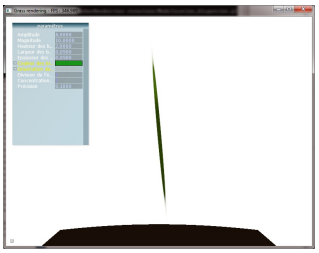
4.1 Modélisation de champ d'herbes

Le champ obtenu par notre modélisation est dépendant de deux aspects : son paramétrage, et ses variations aléatoires. Nous restreignons, dans notre cas, le paramétrage du champ aux paramètres du noyau et de la distribution, les effets des variations et leur ampleur étant gérés par la modélisation.

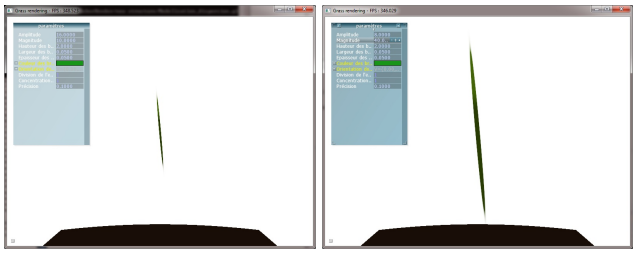
4.1.1 Paramétrage de la modélisation

Notre modélisation permet de modifier l'apparence du champ d'herbe à partir des paramètres du noyau, et des paramètres de la fonction de distribution.

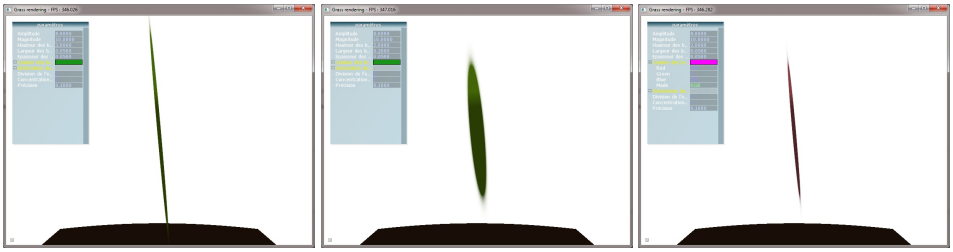
Le noyau gère plus spécifiquement l'aspect visuel des brins d'herbes. Pour cela, nous laissons à l'utilisateur le choix de différents paramètres permettant d'obtenir un large choix d'apparences de brins. Ces différents paramètres, et l'effet de leur augmentation, sont présentés dans la figure 4.1.



(a) témoin



(b) Amplitude inverse (c) Magnitude



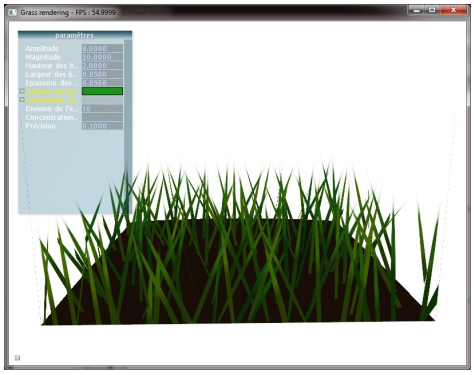
(d) Hauteur (e) Largeur (f) Couleur

FIGURE 4.1 – Augmentation des paramètres du noyau

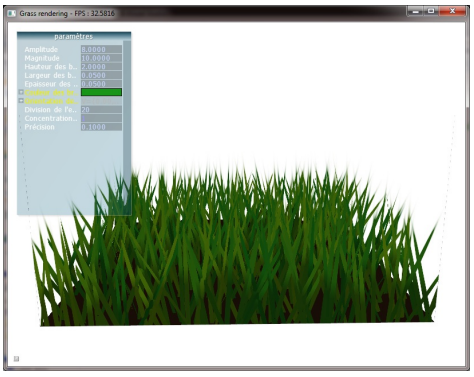
Si les paramètres de hauteur, largeur et couleur sont assez explicites, l'amplitude inverse et la magnitude sont des paramètres mathématiques difficiles

à utiliser. Ces deux paramètres sont hérités de la formule du noyau gaussien : l'amplitude inverse nous permet de contrôler l'échelle du noyau, et la magnitude nous permet de contrôler son intensité.

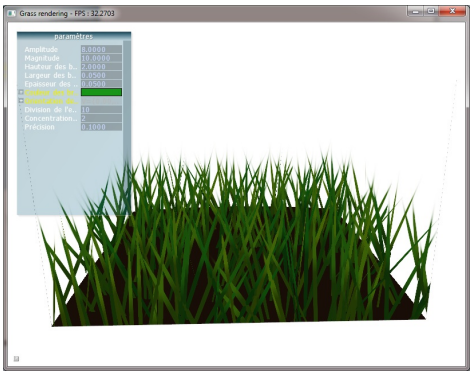
La quantité de brins affichés et leur concentration sont gérées par la fonction de distribution. L'utilisateur peut régler le nombre de cellules, ainsi que le nombre de brins par cellule. Les effets de l'augmentation de ces paramètres sont présentés dans la figure 4.2.



(a) témoin



(b) Nb Cellules



(c) Nb Brins

FIGURE 4.2 – Augmentation des paramètres de la distribution

Ces deux paramètres influencent grandement les performances. L'augmentation du nombre de cellules permet d'augmenter le nombre de brins à l'écran tout en conservant de bonnes performances. Mais un nombre trop important de cellules peut provoquer des artefacts de coupure des brins, et parcourir un très grand nombre de petites cellules reviendrait à échantillonner le rayon pas

à pas.

La concentration de brins par cellule permet de contrebalancer ce problème. Cependant, l'augmentation de ce paramètre peut diminuer fortement les performances, l'évaluation d'un noyau supplémentaire étant couteuse en puissance de calcul. Un équilibre entre ces deux paramètres permet d'obtenir un champ dense avec un rendu fluide.

4.1.2 Variations naturelles

Pour obtenir une apparence plus réaliste, notre modélisation simule des variations naturelles. Comme nous l'avons précisé dans la partie 3, ces variations sont issues de phénomènes physiques complexes difficilement paramétrables. Notre modélisation les génère donc pseudo-aléatoirement.

Notre modélisation définis pour chaque brin une orientation aléatoire, guidée par un vecteur de direction que nous avons fixé allant vers le haut, et les dispose aléatoirement dans les cellules. Mais elle génère aussi, à partir des paramètres utilisateurs, de nouveaux paramètres d'apparence pour le noyau :

- Une couleur légèrement jaunie,
- Une hauteur, plus grande ou plus faible,
- Une largeur, également légèrement plus grande ou plus faible.

La figure 4.3 nous montre l'effet de ces différentes variations sur un exemple.

Ces variations permettent ainsi de donner un champ à l'apparence légèrement hétérogène. Même si cela est possible en ajoutant de nouveaux paramètres, nous n'autorisons pas l'utilisateur à modifier l'ampleur des variations créées dans notre modèle.

4.2 Rendu de la modélisation

Une fois la modélisation réalisée, nous avons cherché à l'utiliser à travers une méthode de rendu volumique générique. Pour commencer les essais, nous avons implémenté une méthode de *ray marching*, permettant théoriquement le rendu le plus précis de l'image. Un problème posé par cette méthode est le

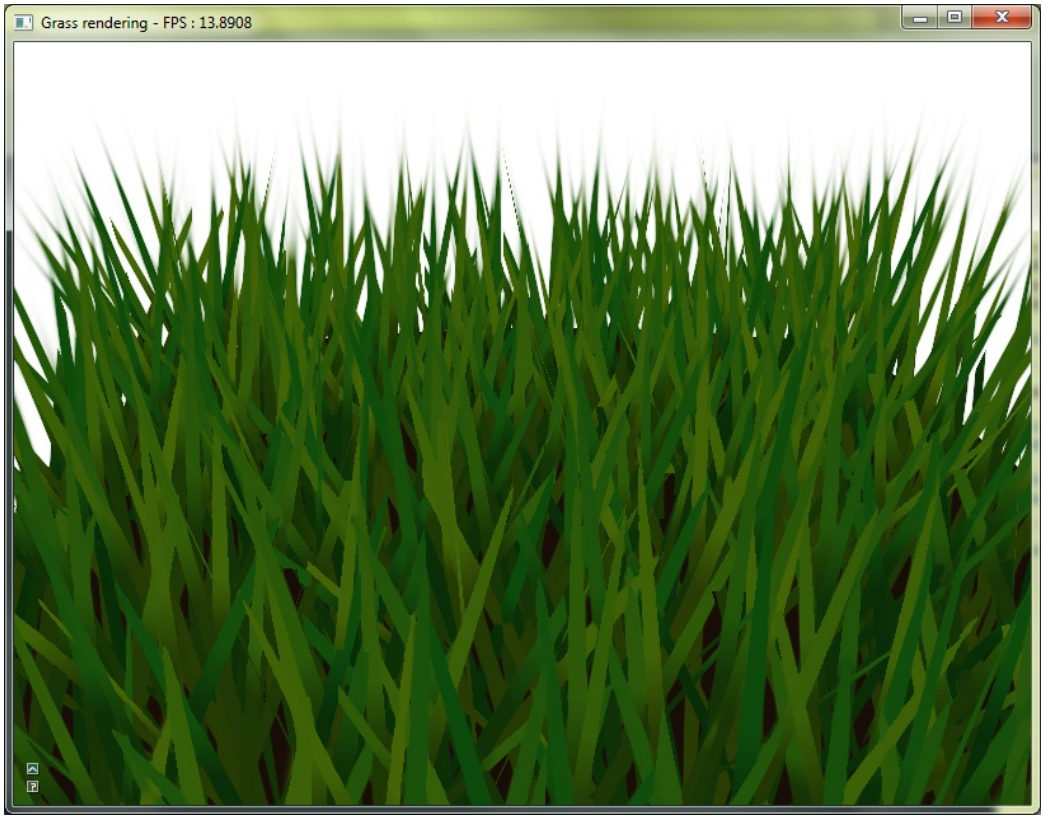


FIGURE 4.3 – Exemple d'applications des variations

nombre de noyaux ne contribuant pas aux échantillons car étant à une trop grande distance d'évaluation.

La première optimisation que nous avons réalisée est la division de l'espace en cellule, pour que chaque échantillon n'évalue que les noyaux se trouvant à sa portée. L'échantillonnage est alors réalisé cellule par cellule. Cependant, cette optimisation continue d'évaluer de trop nombreux échantillons non significatifs à l'intérieur même des cellules.

Notre second optimisation règle ce problème grâce à une recherche d'échantillons significatifs à l'intérieur même des cellules, coupler à une simulation de profondeur pour régler les problèmes de visibilité. La méthode de rendu obtenu permet alors d'obtenir de bien meilleur performances, jusqu'à un résultat

véritablement interactif.

Pour illustrer les améliorations apportées par notre méthode de rendu, nous présentons tout d'abord un des meilleurs résultats obtenu par *ray marching* utilisant la première optimisation citée. Ce résultat est l'un des meilleurs pouvant être obtenu en temps réel, une modélisation plus gourmande et un échantillonnage plus précis provoquant une erreur du pilote graphique.

Nous le comparons au résultat de notre méthode de rendu avec le même paramétrage, avant de présenter des résultats plus complexes que nous n'avons pas pu obtenir avec la méthode de *ray marching*.

4.2.1 Environnement de test

Ces deux méthodes nécessitant un volume, nous utilisons comme modèle de référence un simple cube de dimension 1, dans lequel est exécutée l'évaluation de la modélisation. Le point d'observation est inchangé d'une méthode à l'autre.

Pour comparer les deux méthodes, nous utilisons le paramétrage suivant :

paramètres du noyau	valeur
amplitude inverse	8.0
magnitude	10.0
hauteur	2.0
largeur	0.05
Nombre de cellules par axe	20
Nombre de brin par cellule	1
Pas d'échantillonnage	0.01

TABLE 4.1 – Paramètres de la modélisation

Ce paramétrage nous permet de générer une image avec le *ray marching* avec les performances minimales possibles pour le rendu interactif. Il est possible d'augmenter légèrement le nombre de cellules et de brins mais le risque d'erreur du pilote graphique devient alors trop important.

Le pas d'échantillonnage est ici uniquement appliqué à la méthode de *ray marching*, et est calculé selon les dimensions du cube. Ce pas d'échantillonnage est assez petit pour obtenir un rendu réaliste des brins.

4.2.2 Rendu par *ray marching*

Nous obtenons, avec le paramétrage précédent, l'image présentée par la figure 4.4.

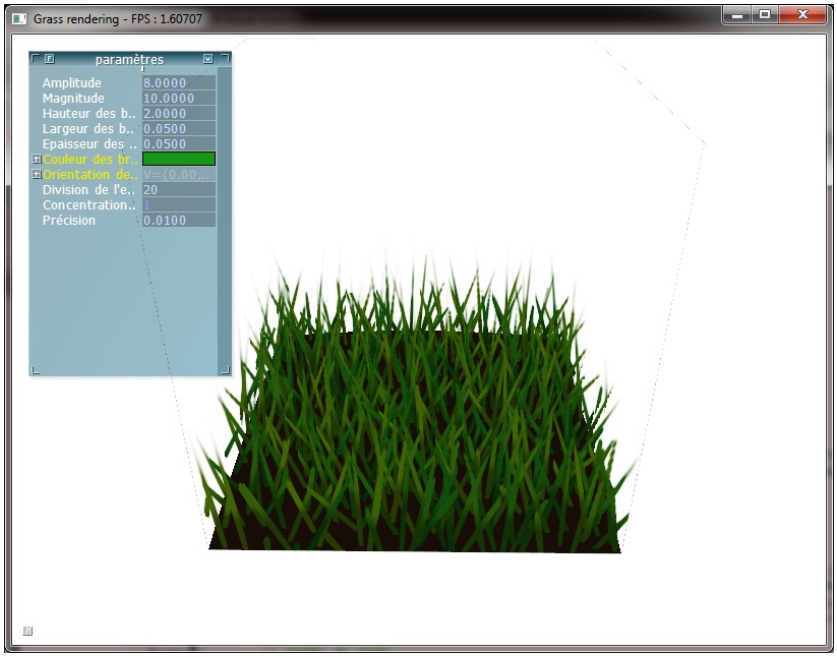


FIGURE 4.4 – Résultat du *ray marching*

Grâce au pas d'échantillonnage faible, nous obtenons une image précise du résultat, mais les performances sont très faibles. Cette méthode, avec les paramètres courants, ne permet d'obtenir qu'une et deux images par seconde. Il est encore possible de modifier interactivement l'apparence, mais avec peu de fluidité.

Cette méthode est particulièrement influencée par le nombre de noyaux par cellule, extrêmement limité dans ce cas. Le nombre de cellules permet cependant d'augmenter significativement le nombre de brins affichés avec une perte minime de performance.

4.2.3 Rendu par la méthode optimisée

En utilisant toujours les paramètres définis précédemment, nous obtenons le résultat présenté par la figure 4.5.

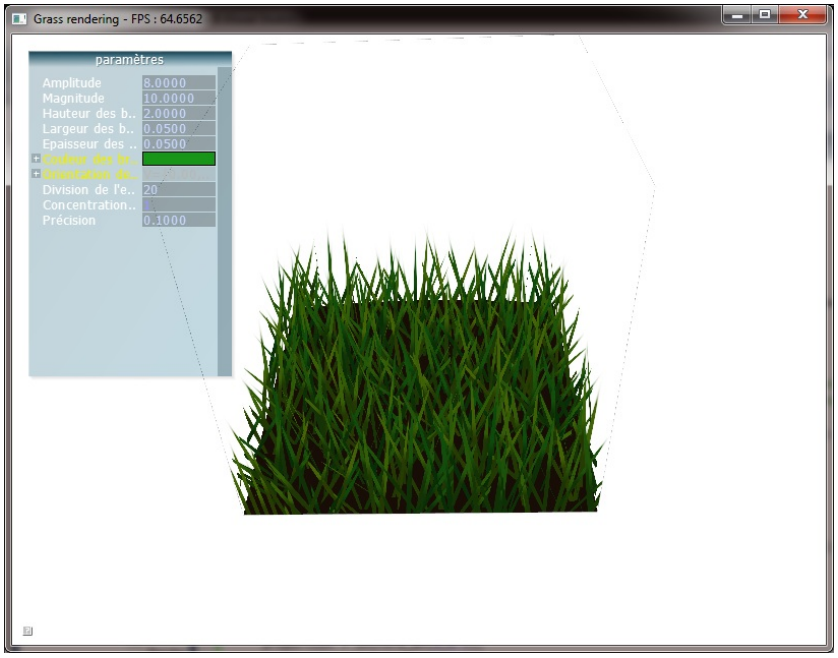


FIGURE 4.5 – Résultat de la méthode optimisée

Contrairement à la méthode de *ray marching*, le rendu obtenu est parfaitement fluide : le résultat est obtenu avec plus de 60 images par seconde. Notre méthode permet effectivement d'améliorer nettement les performances du rendu. Cette méthode peut gérer plus facilement un nombre de brins par cellule plus important, mais tout comme la méthode de *ray marching*, ce paramètre influence grandement les performances.

Un résultat obtenu en utilisant un brin supplémentaire par cellule est présenté par la figure 4.6. Ce paramétrage nous permet d'obtenir un rendu toujours fluide, proche des 30 images par seconde.

Nous repoussons encore un peu le paramétrage, pour tenter de représenter un champ plus grand. Nous diminuons l'échelle de la modélisation en montant l'amplitude inverse à 25.0. Nous augmentons également le nombre de cellules

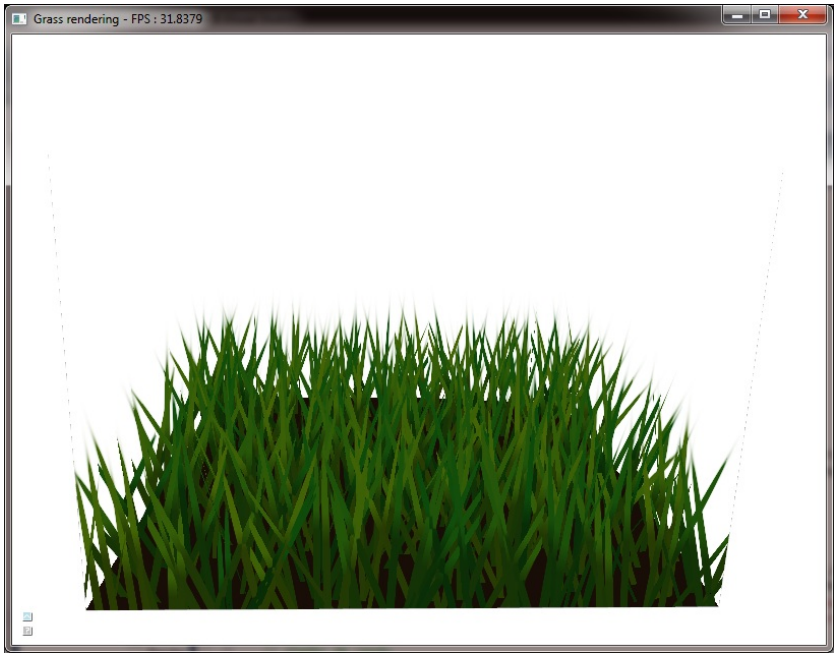


FIGURE 4.6 – 2eme résultat de la méthode optimisée
(2 brins par cellule)

à 625 cellules (soit une grille de 25 par 25) et de brins par cellule à 2 brins par cellule. Le résultat obtenu est présenté par la figure 4.7. Ce résultat est rendu en 14 images par seconde.

Notre méthode de rendu permet donc de plus facilement modéliser interactivement un champ d'herbe avec notre modélisation. Elle permet de d'utiliser et d'afficher cette modélisation de manière fluide et précise.

4.3 Conclusion

Notre modélisation permet effectivement de générer un champ d'herbes paramétrables, modifiable interactivement, et pouvant inclure diverses variations naturelles. Cette modélisation pose les bases de réalisation de futurs noyaux et fonctions de distributions pour obtenir différents types de champs d'herbes ou de détails.

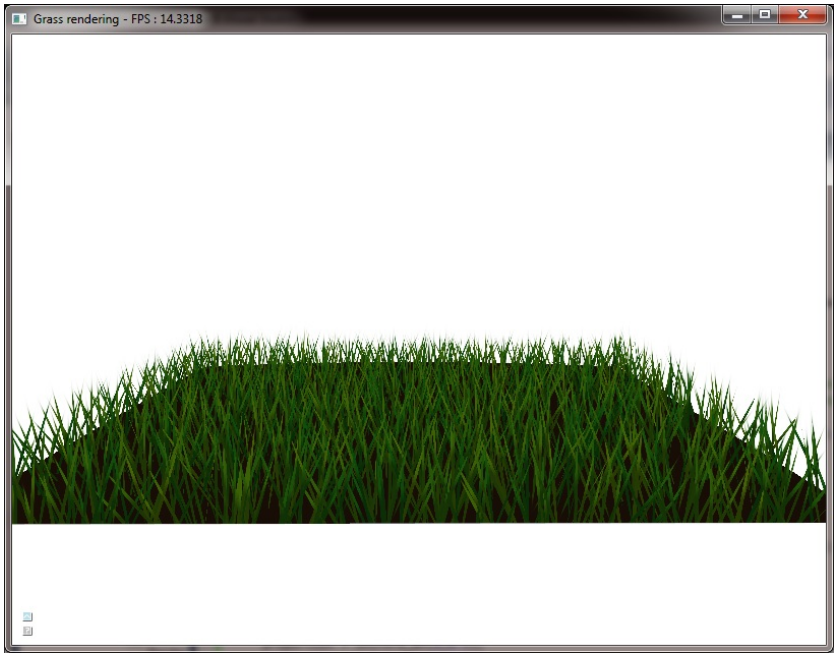


FIGURE 4.7 – 3eme résultat de la méthode optimisée
(brins plus petits et plus nombreux)

La méthode de rendu permet d'utiliser réellement la modélisation interactivement, et cela de manière fluide et précise. Les performances obtenues sont récapitulées dans la table 4.2.

Méthodes	Paramétrage initial	2eme paramétrage	3eme paramétrage
Ray marching	entre 1 et 2 ips 4.4	*	*
Rayon optimisée	≈ 60 ips (4.5)	≈ 30 ips (4.6)	≈ 15 ips (4.7)

* : Erreur : pas de réponse du pilote graphique ;
ips : images par seconde ;
() : image de référence ;

TABLE 4.2 – Performances des méthodes

Chapitre 5

Conclusion et perspectives

5.1 Bilan

Nos contributions sont avant tout des travaux exploratoires, aucun travaux ne présentant de bruit se basant sur des noyaux 3D et permettant de représenter des éléments 3D complexes. Notre objectif était de pouvoir générer procéduralement un champ d'herbes. Cette modélisation devait répondre à différentes problématiques :

- Un temps de création très court,
- Une modélisation interactive,
- Un volume de données stocké minimal.

La modélisation interactive impliquait pour nous la possibilité d'un affichage en temps réel de la modélisation, pour une visualisation immédiate du résultat.

Nous avons apporté une première réponse à ces problématiques par deux contributions. Nous avons tout d'abord développé une modélisation de champ d'herbes procédurale, associée à un rendu par *ray marching*. Nous avons ensuite mis au point une méthode de rendu volumique plus adaptée à cette modélisation, pour améliorer les performances du rendu temps réel.

5.1.1 Modélisation procédurale de champ d'herbes

Notre première contribution concerne la modélisation procédurale d'un champ d'herbes. Grâce à une fonction pseudo-aléatoire, initialement conçu pour créer un bruit procédural, nous avons réussi à créer une fonction permet-

tant de générer différents champs d'herbes.

Cette modélisation est paramétrable, ce qui lui permet de rapidement faire varier l'apparence du champ généré. Nous laissons à l'utilisateur la possibilité de modifier la hauteur et la largeur des brins, ainsi que la couleur des brins.

Basée sur une fonction d'évaluation procédurale, notre modélisation ne représente qu'un coût minime en mémoire et permet de théoriquement représenter des champs à l'infini. Nous ne stockons aucune géométrie hormis le volume englobant, dans lequel la méthode est évaluée, et la fonction d'évaluation paramétrée.

5.1.2 Une méthode de rendu volumique adaptée

Notre deuxième contribution concerne la méthode de rendu optimisée pour la modélisation de l'herbe. Notre modélisation est utilisable avec un rendu par *ray marching*, mais cette méthode de rendu est couteuse en puissance de calcul et n'est pas optimisé en ce qui concerne les éléments très fins, tel que les brins d'herbes.

Nous avons donc proposé avec notre modélisation une méthode de rendu basé sur une recherche d'échantillons significatifs le long du rayon pour chaque noyau. Notre méthode limite ainsi le nombre d'échantillons du rayon au nombre de noyaux présents dans la scène, et ne prend en compte que des échantillons ayant le plus de chance de contribuer a la couleur finale du pixel.

5.2 Perspectives

Notre méthode de génération d'herbe procédurale n'en est actuellement qu'à ses débuts. Ils restent de très nombreuses améliorations à apporter pour obtenir un outil de génération de détails de végétation réalistes, qui puisse représenter une grande variété de végétaux.

5.2.1 Application à un volume générique

Nous avons mis au point notre modélisation en partant d'un simple cube comme support volumique. La première amélioration que nous désirerions mettre en œuvre est l'application de cette modélisation sur une surface ou un volume quelconque, pour qu'elle soit plus facilement et plus rapidement utilisable.

Nos premiers essais d'application de la méthode à un volume générique sont prometteurs, mais les résultats sont perfectibles (figure 5.1). La modélisation et la distribution doivent être retravaillées pour suivre la surface représentée par le volume.

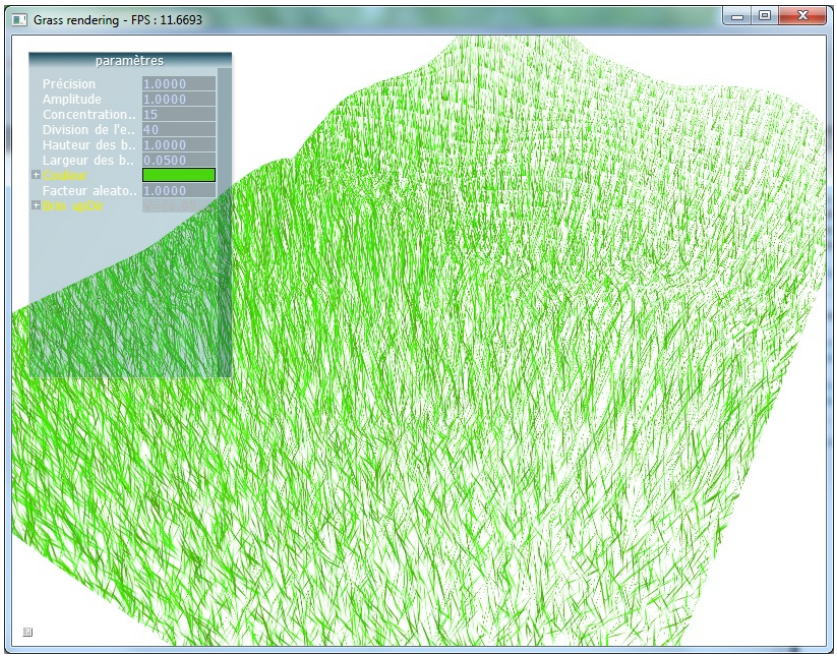


FIGURE 5.1 – Premier essai d'application de la modélisation sur une surface transformée en volume

5.2.2 Gestion du niveau de détail

Notre modélisation n'utilise qu'un seul noyau quelque soit la distance de vue. Ce noyau ne représente pour le moment qu'un unique brin avec préci-

sion, conçu pour le rendu proche. Nous pensons que la création de noyaux plus complexes générant un ensemble de brins est possible.

Ces noyaux multi-brins permettrait de diminuer le nombre de noyau distribuer à moyenne distance, et ainsi d'améliorer les performances. Cette idée peut être étendue à l'utilisation d'un noyau ne représentant plus un ensemble de brin, mais un champ d'herbe continu, qui serait utilisé à longue distance.

5.2.3 Amélioration du noyau

Notre modélisation utilise un noyau issu d'une enveloppe gaussienne pour approximer la forme de l'herbe. Ce noyau nous sert à représenter un brin de manière simplifier et rapide. En nous servant de celui-ci comme base, il est possible d'améliorer grandement la modélisation pour représenter des brins plus complexes.

Le noyau actuel ne prend que des paramètres simples en considération et ne peut donner qu'une forme de brin droite. La première amélioration possible serait la prise en compte de plus de caractéristiques d'un brin. Nous pourrions par exemple introduire la courbure et la torsion du brin dans le noyau, qui seraient elles aussi paramétrable.

Une autre amélioration possible serait l'abstraction de certains paramètres par un paramètre plus générique, facilitant le contrôle de l'apparence du champ. Par exemple, il serait possible de contrôler la hauteur, la largeur et la couleur (ou plus généralement, la croissance) des brins grâce au temps. L'animation et l'orientation des brin pourraient aussi être simplifiées en utilisant un vecteur correspondant à la direction du vent.

A l'instar de Lagae et al. [18] avec le noyau de Gabor, nous pensons qu'il est en plus possible d'associer au noyau une méthode de filtrage, afin d'en améliorer la qualité visuelle. Le filtrage nous permettrait d'améliorer la qualité de l'anti-aliasing de la méthode.

5.2.4 Création d'autres détails

L'objectif que nous nous sommes fixés à l'origine de ce projet est de pouvoir générer des détails de végétation, ce qui inclut de nombreux éléments dif-

férents. Nous nous sommes dans un premier temps restreint à la formation de l'herbe, qui est un élément présent en très grande quantité dans la majorité des scènes de végétation.

Cependant, nous n'avons réalisé la modélisation que d'un seul type d'herbe alors qu'il en existe de très nombreuses espèces. Nous pensons pouvoir adapter notre modélisation pour représenter ces autres espèces, mais aussi d'autres structures végétales telles que les feuilles.

Une autre solution envisagée est d'utiliser des fonctions de transfert de formes pour créer de nouveaux détails. Ces fonctions permettraient de représenter plus facilement des formes complexes telles que les feuilles, voir de paramétrer ses propres formes.

5.2.5 Paramétrage automatique

Notre modélisation utilise un paramétrage manuel du noyau par un utilisateur. Cependant ces paramètres peuvent ne pas être intuitifs, en particulier pour les paramètres mathématiques.

Comme cité dans la partie 2.3.3, de nombreux travaux ce sont intéressés aux propriétés des bruits, plus particulièrement pour reproduire un motif 2D à partir d'une photographie ou d'un exemple. Appliquer cette idée de paramétrage par analyse à notre modélisation est une idée intéressante, qui permettrait de simplifier son utilisation.

Nous pourrions par exemple chercher à reproduire les propriétés spectrales d'un ensemble de photographies d'un carré d'herbe pris sous différents angles de vue.

Bibliographie

[1] T. AKENINE-MÖLLER et E. HAINES : *Real-Time Rendering - Second Edition*, chap. 12.1.3, p. 492–494. A K Peters, 2002.

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

[3] K. BOULANGER : *Real-Time Realistic Rendering of Nature Scenes with Dynamic Lighting*. Thèse de doctorat, University of Rennes, 2008.

[4] A. BOUTHORS, F. NEYRET, N. MAX, E. BRUNETON et C. CRASSIN : Rendu interactif de nuages réalistes. *In 20èmes Journées de l'Association Française d'Informatique Graphique, AFIG 2007, November, 2007*, p. 183–195, Marne la Vallée, France, nov. 2007. AFIG.

[5] P. DECAUDIN et F. NEYRET : Volumetric billboards. *Computer Graphics Forum*, 28(8):2079–2089, 2009.

[6] A. R. FULLER, H. KRISHNAN, K. MAHROUS, B. HAMANN et K. I. JOY : Real-time procedural volumetric fire. *In Proceedings of the 2007 symposium on Interactive 3D graphics and games*, I3D '07, p. 175–180, New York, NY, USA, 2007. ACM.

[7] B. GALERNE, A. LAGAE, S. LEFEBVRE et G. DRETTAKIS : Gabor noise by example. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2012)*, 31(4):73 :1–73 :9, July 2012. To appear.

[8] G. GILET : *Ajout de détails visuels complexes en temps réel pour la synthèse d'images*. Thèse de doctorat, Université de Strasbourg, Sep 2009. Ecole doctorale MSII.

[9] G. GILET et J.-M. DISCHLER : An image-based approach for stochastic volumetric and procedural details. *Computer Graphics Forum*, 29(4):1411–1419, Jun 2010.

[10] G. GILET, J.-M. DISCHLER et D. GHAZANFARPOUR : Multiple kernels noise for improved procedural texturing. *The Visual Computer*, 28:679–689, 2012. 10.1007/s00371-012-0711-2.

[11] G. GILET, J.-M. DISCHLER et L. SOLER : Procedural descriptions of anisotropic noisy textures by example. *In Eurographics*, 2010. Short paper.

[12] S. GUERRAZ, F. PERBET, D. RAULO, F. FAURE et M.-P. CANI : A procedural approach to animate interactive natural sceneries. *In 16th International Conference on Computer Animation and Social Agents, CASA 2003, May, 2003*, p. 73–78, Rutgers University, New Brunswick, NJ, Etats-Unis, 2003. IEEE.

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

[14] A. KENSLER, A. KNOLL, P. SHIRLEY, A. KENSLER, A. KNOLL et P. SHIRLEY : Better gradient noise, 2008.

[15] A. LAGAE et G. DRETTAKIS : Filtering solid Gabor noise. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2011)*, 30(4):51 :1–51 :6, July 2011.

[16] A. LAGAE, S. LEFEBVRE, R. COOK, T. DEROSE, G. DRETTAKIS, D. EBERT, J. LEWIS, K. PERLIN et M. ZWICKER : A survey of procedural noise functions. *Computer Graphics Forum*, 29(8):2579–2600, December 2010.

[17] A. LAGAE, S. LEFEBVRE, G. DRETTAKIS et P. DUTRÉ : Procedural noise using sparse gabor convolution. *In ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, p. 54 :1–54 :10, New York, NY, USA, 2009. ACM.

[18] A. LAGAE, S. LEFEBVRE et P. DUTRÉ : Improving Gabor noise. *IEEE Transactions on Visualization and Computer Graphics*, 17(8):1096–1107, August 2011.

[19] J. LENGYEL, E. PRAUN et A. FINKELSTEIN : Real-time fur over arbitrary surfaces. *In 2001 ACM Symposium on Interactive 3D Graphics*, p. 227–232, 2001.

[20] P. MÜLLER, P. WONKA, S. HAEGLER, A. ULMER et L. VAN GOOL : Procedural modeling of buildings. *In ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, p. 614–623, New York, NY, USA, 2006. ACM.

[21] Y. I. H. PARISH et P. MÜLLER : Procedural modeling of cities. *In Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, p. 301–308, New York, NY, USA, 2001. ACM.

[22] K. PELZER : Rendering countless blades of waving grass. *In GPU Gems*, p. 107–121. Addison-Wesley, march 2004.

[23] F. PERBET et M.-P. CANI : Animating prairies in real-time. *In Symposium on Interactive 3D Graphics, SI3D 2001, March, 2001*, p. 103–110, Chapel Hill, NC, Etats-Unis, mars 2001. ACM-SIGGRAPH, ACM.

[24] K. PERLIN : An image synthesizer. *In Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '85, p. 287–296, New York, NY, USA, 1985. ACM.

[25] K. PERLIN : Improving noise. *In Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, p. 681–682, New York, NY, USA, 2002. ACM.

[26] K. PERLIN et F. NEYRET : Flow noise. *In Siggraph Technical Sketches and Applications*, p. 187, Aug 2001.

[27] K. PERLIN, M. OLANO, J. C. HART, W. HEIDRICH et B. MARK : chapter 2 - noise hardware. *In Real-time shading language*. SIGGRAPH course 36, 2002.

[28] J. PERRET : *Modélisation d'environnements urbains virtuels*. Thèse de doctorat, December 2006.

[29] A. PEYRAT, O. TERRAZ, S. MERILLOU et E. GALIN : Generating vast varieties of realistic leaves with parametric 2gmap l-systems. *The Visual Computer*, 24:807–816, 2008. 10.1007/s00371-008-0262-8.

[30] P. PRUSINKIEWICZ et A. LINDENMAYER : *The Algorithmic Beauty of Plants*, chap. 2. Springer-Verlag, 1990.

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

[32] O. TERRAZ, G. GUIMBERTEAU, S. MÉRILLOU, D. PLEMENOS et D. GHAZANFARPOUR : 3gmap l-systems : an application to the modelling of wood. *The Visual Computer*, 25:165–180, 2009. 10.1007/s00371-008-0212-5.

[33] D. WHATLEY : Toward photorealism in virtual botany. *In GPU Gems 2*, p. 7–25. Addison-Wesley, march 2005.