Représentation, modélisation et génération procédurale de terrains

Jean-David Génevaux

Thèse

pour obtenir le grade de

Docteur de l'Université Lumière Lyon 2
en informatique

réalisée sous la direction du Pr. Éric Galin

et co-encadrée par Dr. Éric Guérin

REPRÉSENTATION, MODÉLISATION ET GÉNÉRATION PROCÉDURALE DE TERRAINS

Jean-David Génevaux

Thèse soutenue le : 3 Septembre 2015

Composition du jury :

Encadrants :	
M. Éric Galin,	Pr. à l'Université Lumière Lyon 2
M. Éric Guérin,	MCF à l'INSA de Lyon
Rapporteurs :	
M. Loïc Barthe,	Pr. à l'Université Paul Sabatier de Toulouse
M. Stéphane Mérillou,	Pr. à l'Université de Limoges
Examinateurs :	
M. Jean-Jacques Bourdin,	Pr. à l'Université Paris 8
M. Jean-Michel Dischler,	Pr. à l'Université de Strasbourg
M. Adrien Peytavie,	MCF à l'Université Claude Bernard Lyon 1

Hodie Mihi, Cras Tibi

Thèse réalisée sous la direction du Pr. Éric Galin
et co-encadrée par Dr. Éric Guérin.

Travaux effectués à Lyon entre Septembre 2011 et Mai 2015
au Laboratoire d'InfoRmatique en Images et Systèmes d'information
dans le cadre de l'École Doctorale InfoMaths.

À ceux qui cherchent et qui trouvent
… et à tous ceux qui cherchent sans trouver.

Abstract

This PhD (entitled "Representation, modelisation and procedural generation of terrains") is related to movie and videogames digital content creation, especially natural scenes.

Our work is dedicated to handle and to generate landscapes efficently. We propose a new model based on a construction tree inside which the user can handle parts of the terrain intuitively. We also present techniques to efficently visualize such model. Finally, we present a new algorithm for generating large-scale terrains exhibiting hierarchical structures based on their hydrographic networks : elevation is generated in a broad compliance to water-tansport principles without having to resort on costly hydraulic simulations.

Keywords : terrain models, procedural modeling, procedural generation, data structure, terrains

Résumé

Cette thèse (qui a pour intitulé « Représentation, modélisation et génération procédurale de terrains ») a pour cadre la génération de contenus numériques destinés aux films et aux jeux-vidéos, en particulier les scènes naturelles.

Nos travaux visent à représenter et à générer des terrains. Nous proposons, en particulier, un nouveau modèle de représentation qui s'appuie sur un arbre de construction et qui va permettre à l'utilisateur de manipuler des morceaux de terrain de façon intuitive. Nous présentons également des techniques pour visualiser ce modèle avec un maximum d'efficacité. Enfin nous développons un nouvel algorithme de génération de terrains qui construit de très grands reliefs possédant des structures hiérarchiques découlant d'un réseau hydrographique : le relief généré est conforme aux grands principes d'écoulement des eaux sans avoir besoin d'utiliser de coûteuses simulations d'érosion hydrique.

Mots clés : modèles de terrains, modélisation procédurale, génération procédurale, structures de données, terrains

Publications

Journaux

[141] Jean-David Génevaux, Éric Galin, Adrien Peytavie, Éric Guérin, Cyril Briquet, François Grosbellet, Bedřich Beneš
Terrain Modeling from Feature Primitives
Computer Graphics Forum – CGF
2015, Wiley-Blackwell

[140] Jean-David Génevaux, Éric Galin, Éric Guérin, Adrien Peytavie, Bedřich Beneš
Terrain Generation Using Procedural Models Based on Hydrology
Transaction on Graphics – TOG
2013, vol. 32, num. 4, p. 143 :1–143 :13, ACM
Présenté à *SIGGRAPH* 2013, Anaheim, USA

Conférences

[142] Jean-David Génevaux, Éric Galin, Adrien Peytavie, Éric Guérin, Cyril Briquet, François Grosbellet, Bedřich Beneš
Modélisation de terrains par primitives
Proc. of the Journées de l'Association Française d'Informatique Graphique – AFIG
Reims, France, 2014, IRIT Presse
⋆ Prix du 2ème meilleur papier ⋆
Également présenté à *Paris ACM SIGGRAPH* 2015

[139] Jean-David Génevaux, Éric Galin, Éric Guérin, Adrien Peytavie, Bedřich Beneš
Génération procédurale de rivières et de terrains
Proc. of the Journées de l'Association Française d'Informatique Graphique – AFIG
Calais, France, 2012, IRIT Presse
⋆ Prix du 2ème meilleur papier ⋆
Également présenté à *Paris ACM SIGGRAPH* 2013
Également présenté aux *rencontres LIMA2 / ARC6* 2013
Également présenté à la *journée des thèse du LIRIS* 2013

Remerciements

Gratitude is best and most effective when it does not evaporate itself in empty phrases.

Foundation and Empire — Isaac Asimov

Bonjour à toi cher lecteur ! Il y a de fortes chances que cette section soit la première que tu lises (et probablement la dernière, je n'en doute pas). Mais commençons sans tarder… Je remercie :

Mes encadrants :

- Éric Galin : merci d'avoir accepter de diriger ma thèse ! Je me souviens en particulier de nos premiers échanges de mails et de notre première rencontre dans les locaux du LIRIS, le 11 juillet 2011. Je te remercie de m'avoir laissé une grande liberté sur le sujet de mes recherches [1]. J'ai appris de nombreuses choses à ton contact : la curiosité et la passion nécessaires pour explorer des pistes et des thèmes de manière originale, la nécessaire confrontation des opinions, le suivi de projets (à Gamagora) ou encore le soucis du « beau » (et du clair !) lors de la rédaction d'articles scientifiques.

- Éric Guérin : merci également d'avoir encadré ces travaux ! Te souviens-tu de notre virée nocturne à Paris, après la présentation à Paris ACM SIGGRAPH ? Nous avions soumi à SIGGRAPH sans grand espoir et nous nous attendions à un probable rejet. Ton ami, Alex, nous a enjoint de profiter de ce que nous avions déjà mais prophétiquement, il avait mentionné le fait que notre article pourrait être accepté. C'était un véritable plaisir de travailler avec toi et ta bonne humeur pendant toutes ces années. Bonne chance pour ton HDR [2] !

- Adrien Peytavie : si tu n'as pas officiellement le titre d'encadrant, c'est tout comme. Au laboratoire, tu as toujours été présent pour répondre à mes questions. Mais il n'y a pas que le travail et je garderai en mémoire de nombreux souvenirs de notre *roadtrip* aux États-Unis ; en particulier quand nous avons joué des sommes folles aux casinos de Las Vegas après plusieurs heures d'hésitation [3] !

1. J'ai vérifié, le document original de l'école doctorale était intitulé « Modélisation de mondes virtuels ».
2. Tu vois, moi aussi je sais faire des blagues pas drôles.
3. Que le lecteur soit rassuré, nous n'avons dépensé que 10$ chacun, soit la somme minimum pour entrer sur la table... et nous les avons perdu illico !

Les autres membres du jury :

- Loïc Barthe et Stéphane Mérillou : vous avez accepté de rapporter ma thèse et je vous en remercie. Votre sérieux et votre expertise – mais aussi votre gentillesse – m'impressionnent à chaque fois et je suis honoré de confronter ce manuscrit à votre précieuse relecture. À chaque fois que cela a été possible, ce fut un réel plaisir de vous voir et de pouvoir discuter avec vous en conférence [4].

- Jean-Michel Dischler et Jean-Jacques Bourdin : vous m'avez fait l'amitié, l'un et l'autre de lire mon manuscrit. Je remercie Jean-Michel d'avoir accepter de relire mes travaux ainsi que de m'avoir enseigné l'informatique graphique alors que tu as déjà dirigé la thèse d'un autre Génevaux, qui lui aussi ne s'est pas privé d'écrire un pavé de plusieurs centaines de pages. Mes plus sincères remerciements vont également à Jean-Jacques, avec lequel j'avais évoqué une participation à mon jury il y a de nombreux mois déjà. C'est une chance que j'ai répondu complètement à côté de la plaque à ta question durant ma présentation à l'AFIG 2012 : c'est pour cette raison que nous avons commencé à échanger quelques mots à la pause suivante ! Depuis, nous avons eu le plaisir de nous revoir et de discuter de jeux de rôles [51] [5].

Mes collègues de travail :

- Le bureau du LIRIS-1 (passé et présent) : **Elsa**, **Jérémy**, **François**, Aurélien, Camille, Grégoire, Abdoulaye, Gérémy et Matthieu. J'ai passé de superbes années en votre compagnie : ce bureau était vraiment celui de l'amitié. Je signe tout de suite pour pouvoir partager à nouveau un espace de travail avec vous !

- Les amis du LIRIS-1 (passé et présent) : Lucian, Clément, Matthieu, Jeremy, Loïc, Lise, Anaïs, Lucille, Marie-Neige, Joseph, Karolina et Hélène. Grâce à vous, toujours de l'animation au Nautibus !

- David « The Dude » Coeurjolly : pour les nombreuses discussions autour de LaTeX, Fourier, la GeoDis, SIGGRAPH…, mais aussi pour connaître les répliques de ce monument cinématographique qu'est Top Gun « *Son, your ego is writing checks your body can't cash.* ». J'espère un jour écrire un article avec toi.

- Toute l'équipe Géomod (et en particulier Raphaëlle et Samir qui l'ont dirigée ces dernières années). Merci également à Alexandre, Florence, Julien qui ont accepté que je les aide à donner des cours dans leurs modules.

4. Omettons de parler de la soirée canadienne (SIGGRAPH 2013) ou du spectacle des danses traditionnelles alsaciennes (EG 2014).

5. Fait attention ! Depuis que tu m'as soulevé en m'attrapant par le col au repas de gala d'Eurographics, je crois que les gens ont peur de toi !

- Amélie, Matthieu, Gurpeet, Adrien, Damien, Jonathan, André, Florent, Gaël, Alex, Xavier, Cyril, Donatien, Nicolas, Guillaume, Julie, Elodie, Jean-Claude, Eliane, Thierry, Jean-Philippe, Pierre-Marie, Stéphanie, Erwan, Christine, Gilles, Vincent, Victor, Tristan, ... qui ont égayé mes journées au sein du laboratoire ou en dehors. Je souhaite également remercier Brigitte et le reste du personnel administratif qui m'ont aidé de nombreuses fois avec le sourire.

- Pascal Schreck (ainsi que Simon Thierry et Julien Narboux) : merci de m'avoir fait découvrir la recherche académique ! Sans vous, je ne serais jamais allé faire une thèse à Lyon.

- La communauté de l'informatique graphique française et internationale : beaucoup de gens vraiment cools que j'ai découvert. Je salue Cyril, Thomas, Maxime, Florian, Boris, Ulysse, Quentin, Evans, Nicolas, Guillaume, Kenneth, Lionel, Remi, Manuel, Simon, Nicolas, Nicolas, Richard, Élie, Jonathan, Céline, Cécile, Arnaud, Thomas, Andrew, Pierre, Pierre, David, Gilles, Sebastien, Romain, Venceslas, Georges-Pierre, Benoît, Jean-François et Bedřich.

- Les partenaires sans qui rien n'aurait été possible : la boulangerie Régis Grand et les Burgers à Papa. Je n'oublie pas les tumblr "*Ciel mon Doctorat !*" [7] et "*Je remercie, tu remercies, nous remercions*" [182] ainsi que les web-BD "*PhD Comics*" [68], "*xkcd*" [250], "*explained xkcd*" [244], "*Carnet de thèse / Le bureau 14 de la Sorbonne*" [323], "*Vie de thésarde / Vie de jeune docteure*" [207], "*Tu mourras moins bête (mais tu mourras quand même)*" [249] et le blog de Boulet [329] .

En dehors du travail :

- Sundown et JOC : cette thèse n'aurait pas pu se dérouler sans mes assistantes sociales qui ont toujours accepté d'écouter mes petits malheurs !

- La Team des Marmousets (Meuhoua, Eurok, mcgrill) et les différents canards lyonnais : ils ont partagé de très nombreuses et très tardives soirées ainsi que de superbes fous rires et mémorables discussions.

- Les Barons du Vertex (Karganys, Mierre, Dodolf, L0ur5) : j'ai eu le plaisir de discuter informatique et jeux-vidéo avec vous de nombreuses fois ; nous avons créé un club de *gentlemen*, c'est dire ! J'espère que les baronnades continueront encore fort longtemps.

- Mes parents : ils détiennent probablement le record du soutien le plus long dans le temps (environ 27 ans de soutien inconditionnel) en m'encourageant toujours à faire ce que je souhaitais. Merci Papa, merci Maman ! Désolé de ne pas vous avoir téléphoné plus souvent.

- Olivier : je t'ai dit de nombreuses fois que tu es un bien meilleur chercheur que moi (et je continue à le penser) ! Au cours de cette thèse, j'ai eu l'occasion de lire *La Horde du Contrevent* de Alain Damasio[6] et un passage du livre m'a beaucoup marqué :

  > « Le jour où il m'a consacré, Te Jerkka m'a dit que **je n'étais pas le meilleur** combattant-protecteur. Que je ne le serais **jamais**. Mais que c'était aussi pour ça qu'il m'avait choisi : parce que j'avais le ***strerf***, le combat intérieur de celui qui se sait en dessous. "Le meilleur tu deviendras à force de pas l'être, et de toi battre pour surmonter ce sentiment". » ***La Horde du Contrevent* — Alain Damasio**

  En ayant ressenti ce *strerf* ces quatre ans, j'espère avoir fait une thèse aussi bien que la tienne. Tu es un modèle pour moi et je n'ai qu'une hâte : travailler enfin avec toi !

- Hélène : depuis plus d'un an, tu m'as aussi supporté (dans les deux sens du termes). Je t'avais prévenu que la rédaction serait longue et parfois que mon moral serait à zéro. Je te remercie de m'avoir épaulé durant cette période objectivement [412] difficile. Je finirai sur ces quelques mots d'Oldelaf :

  > Restes là ♪
  > Pauvre pomme ♫
  > Ca viendra ♪
  > Attends un peu le Syndrome de Stockholm ♫
  >
  > ***Stockholm (***de l'album ***Dimanche)* — Oldelaf**

Les oubliés[7] :

Si vous lisez ce chapitre, il est probable que vous cherchiez à tromper votre ennui ou que vous cherchiez désespérément votre nom. Je vous invite donc à remplir la ligne suivante avec votre nom :

- Enfin, je remercie plus que tout ..... pour tout ce que je lui dois et lui dédie cette thèse.

6. Si tu n'as pas lu, lecteur, je te le recommande vivement.
7. En reprenant une idée de Antoine Taveneaux [373] trouvée sur le tumblr "*Je remercie, tu remercies, nous remercions*" [182].

Table des matières

1

Introduction

Far better is it to dare mighty things,
to win glorious triumphs, even though checkered by failure…
than to rank with those poor spirits who neither enjoy nor suffer much,
because they live in a gray twilight that knows not victory nor defeat.

— Theodore Roosevelt

Ils avaient déjà raison... il y a presque 50 ans ! Dans une publicité [171] de 1967 – au ton légèrement anxiogène – une célèbre compagnie d'ordinateurs déclame :

Machines should work. People should think. **IBM**

Hélas, il reste encore du chemin à parcourir en informatique graphique avant d'avoir à disposition des outils et des méthodes qui automatisent la création de contenu artistique.

FIGURE 1 : La création d'une scène virtuelle complexe se fait en plusieurs étapes : il faut modéliser le relief et les différents objets du décors. Il est également nécessaire de texturer et d'animer les scènes ainsi que de paramétrer l'éclairage. Tous les outils utilisés doivent offrir une liberté artistique totale aux différents infographistes.

La création numérique

Dans cette thèse, nous allons nous concentrer sur une problématique précise : celle des **scènes naturelles**, qui occupent une place importante dans la plupart des films et des jeux-vidéos. Comment améliorer et simplifier la création de mondes virtuels convaincants ?

La construction d'une scène numérique fait intervenir quatre éléments d'informatique graphique distincts (Fig. 1) : **la modélisation** des objets 3D, **l'application de textures et de matériaux** sur ces objets et **l'animation** de ces objets. Une fois tous les objets prêts et placés, il faut régler les paramètres qui vont influencer **le rendu** de la scène (caméra, lumières, atmosphère, effets météo, etc...). C'est la combinaison de ces quatre éléments qui donnera toute leur richesse visuelle aux scènes virtuelles.

Une stratégie classique pour construire un environnement virtuel consiste à **décomposer hiérarchiquement la création** de la scène. On construit les éléments les plus gros en premier, puis on s'intéresse aux éléments de plus en plus petits. L'élément le plus structurant d'un paysage étant son relief, c'est généralement le terrain qui est modélisé en premier. C'est un élément difficile à créer, car il doit rester unique (la réutilisation d'un morceau de terrain se remarque immédiatement). Après avoir modélisé le relief, les infographistes créent une bibliothèque de décorations (*props*), comprenant par exemple des arbres ou des rochers. Suit le placement manuel ou semi-automatique de ces objets dans la scène 3D. Enfin, la finalisation de la scène dépend du réglage fin des paramètres globaux comme l'éclairage ou la météo, et peut comprendre le positionnement de très petits objets (traces, touffes d'herbes) correspondant soit à de la géométrie, soit à des textures.

FIGURE 2 : GRAND THEFT AUTO V, sorti en 2015 sur PC, est un succès commercial et critique. Son développement a duré 6 ans, coûte environ 265 millions de dollars et plus de 1000 personnes ont contribué à sa conception. Seuls quelques très gros studios de développement peuvent se permettre de se lancer dans un projet de telle envergure. Le développement de nouveaux outils plus intuitifs ainsi que de nouvelles technologies plus performantes permettrait à des studios plus petits de se lancer dans la création de jeux aussi ambitieux en termes de contenu.

Défis en informatique graphique

En 2012, Johan ANDERSSON (directeur technique de DICE) dans un travail collaboratif [3], propose 5 challenges majeurs en informatique graphique pour la création de jeux-vidéo dans le cadre du rendu temps-réel. Parmi ces cinq challenges, deux relèvent du champ d'investigation de cette thèse : **les coûts de production** et **la scalabilité** (*i. e.* la possibilité de passer à l'échelle ou *scaling*).

Les temps de développement d'un jeu dépendent avant tout du temps passé à la création de contenu 2D et 3D (Fig. 2). Les principales pistes pour gagner du temps concernent la conception de modèles multi-échelles pour représenter la géométrie, la numérisation facilitée et accessible à tous de contenus réels, et aussi l'utilisation de génération procédurale de textures et de géométrie.

En ce qui concerne la scalabilité, le souhait est d'évoluer vers des scènes de plus en plus grandes (où coexistent des détails de l'ordre du millimètre et des objets de plusieurs kilomètres). De plus, chaque modèle 3D est aujourd'hui rendu sur des écrans de résolution de plus en plus grande, ce qui a des répercutions sur les temps de calcul. On recherche donc des techniques, des algorithmes et des *pipeline* de production, capables de passer à l'échelle de manière linéaire.

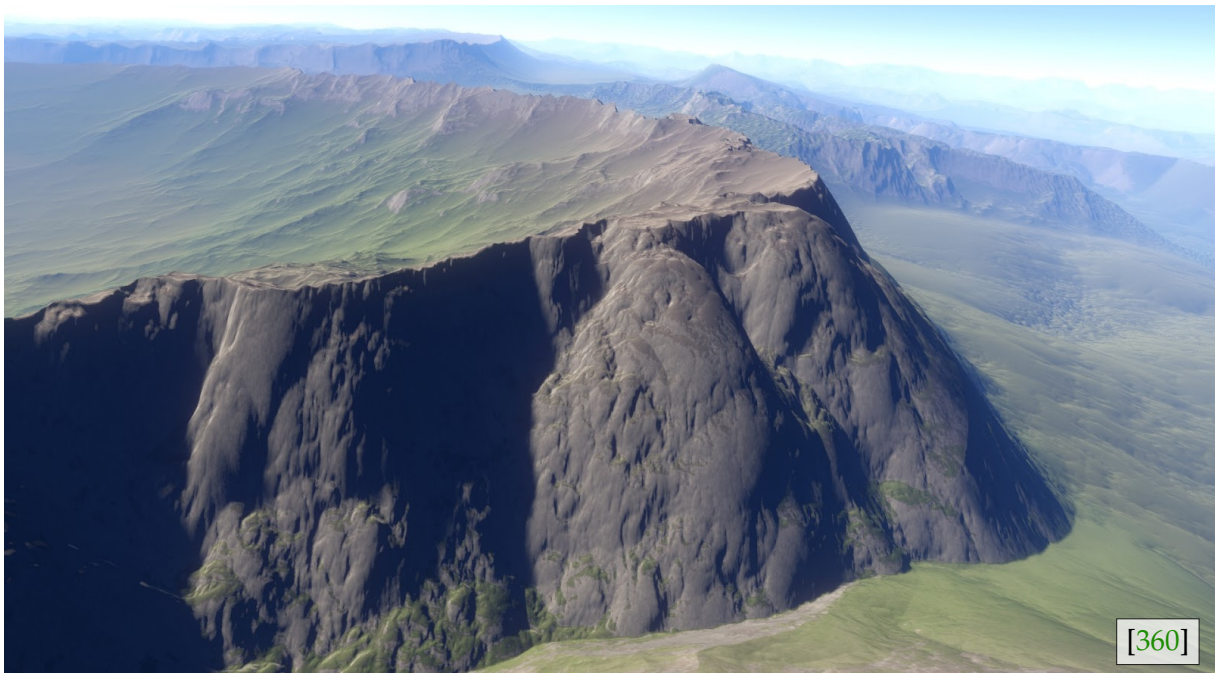


FIGURE 3 : La génération et la visualisation d'un relief demandent des techniques très spécifiques. Seuls des systèmes *scalables* permettent, comme ici, de représenter des planètes entières.

Tout utilisateur de jeu veut pouvoir explorer des mondes virtuels aussi étendus que possible... mais il n'est pas facile de garantir une qualité graphique constante sur l'ensemble de ce paysage !

Un exemple de système intelligent et *scalable* pour la gestion d'un terrain est utilisé dans BATTLEFIELD 3 et a été présenté par DICE à la GDC 2012 [409]. Pour générer des paysages vraiment immenses (*i. e.* des corps planétaires), il est souvent nécessaire de passer par des moteurs spécialisés comme OUTERRA [186] ou le I-NOVAE ENGINE [360] (Fig. 3).

La génération procédurale

Une des solutions est de passer par des méthodes procédurales. **La génération procédurale de contenu** (*Procedural Content Generation* ou PCG) est une famille de méthodes où, au lieu de stocker des objets 3D, on stocke les paramètres d'un algorithme de construction. Ces méthodes ont été très utilisées dans les années 70-80, à l'époque où les ordinateurs avaient peu de mémoire. Pour créer des univers très étendus, on s'appuyaient généralement sur ces méthodes qui utilisent beaucoup la notion d'aléatoire.

Les utilisateurs de ces méthodes procédurales, comptent parmi eux ces artistes-programmeurs que sont les *demomakers* (Fig. 4). Le *demomaking* est une passion à laquelle s'adonnent souvent lors de concours, des programmeurs graphiques qui rivalisent pour créer des scènes, les plus spectaculaires visuellement possibles [374] avec une contrainte de taille-mémoire du programme (1kB, 4kB, *etc*).

FIGURE 4 : Cette scène a été entièrement réalisée de manière procédurale c'est-à-dire par un programme informatique et sans aucun modèle 3D créé par un infographiste.

Aujourd'hui, les ordinateurs ont souvent plusieurs tera-octets de mémoire de stockage et plusieurs giga-octets de mémoire vive. Il n'est pas rare que les fichiers de données des jeux-vidéo ne soient mêmes plus compressés (GTA V, sorti en 2015 sur PC, est livré sous la forme de 7 DVD d'installation soit près de 65 Go de données). Pourquoi alors vouloir générer des résultats si il n'y a aucun problème de stockage ?

Si ces considérations sur le coût-mémoire sont devenues moins importantes, les méthodes procédurales ont d'autres avantages [1]. Elles sont capables d'exprimer une multitude de variations d'un même objet, elles rendent la création numérique davantage automatique, et avec elles, il est possible de représenter des objets d'une taille et d'une résolution quasi-infinie.

Tout n'est alors pas négatif et de nouveaux travaux émergent depuis quelques années ! Aujourd'hui, si la plupart des outils de création utilisés dans l'industrie se basent sur la création manuelle, quelques-uns s'appuient sur des technologies procédurales. SUBSTANCE [2] permet par exemple de peindre des textures procédurales sur des objets ; VOXEL FARM [67] s'appuie sur de nombreux algorithmes de génération ; SPEEDTREE [396] réussit à décorer n'importe quel paysage avec une flore réaliste et HOUDINI [355] est un logiciel dédié à la création d'effets spéciaux à l'aide de procédures.

Notre objectif est donc de **générer des scènes naturelles** en nous appuyant sur des **méthodes de génération procédurales**. Mais pour que ces méthodes soient utilisables dans l'industrie, il est nécessaire de proposer un contrôle intuitif à l'utilisateur.

> *When synthesizing worlds of our own, **complexity equals work**. This work can be on the part of the programmer/ artist or the computer ; in fact it will always be some of each. But there is a balance to be struck, and personally I prefer to have the computer do most of the work. Usually, it seems to have nothing better to do, while I generally can find other mischief to make.*
>
> **Forest Kenton MUSGRAVE**

1. Une définition plus précise de ce que nous appelons les modèles « procéduraux » est donnée Section 2.2.4.

Plan de la thèse

Notre thèse s'articule autour de six chapitres.

Dans le Chapitre 2 nous présentons un état de l'art qui recense les différents modèles possibles de représentations de terrains. Nous détaillons également toutes les méthodes de génération, de simulation et d'édition de reliefs. Au cours de cet état de l'art, nous proposons d'utiliser un nouveau modèle de comparaison de méthodes, sous forme de diagramme radar, qui permet d'analyser les forces et faiblesses de chaque représentation.

Le Chapitre 3 présente un nouveau modèle de représentation qui s'appuie sur un arbre de construction, où chaque feuille correspond à un morceau de relief, et où chaque nœud permet d'agréger différents morceaux de terrains. Chaque arbre définit une fonction d'élévation et permet de combiner modèles procéduraux et vectoriels. Ce modèle est capable de représenter de très grands paysages manière intuitive et à faible coût-mémoire. Il permet également de gérer l'instanciation et comporte un système de niveaux de détails.

Le Chapitre 4 propose plusieurs techniques de visualisation de ce modèle. Une technique de raffinement de maillage permet de visualiser correctement les fonctions de terrains multi-matériaux. Il est également possible de visualiser notre fonction d'élévation directement par lancer de rayons. Cette technique de *sphere tracing* s'appuie sur le calcul des bornes de Lipschitz de nos différentes fonctions d'élévation.

Le Chapitre 5 développe un algorithme de génération procédurale, capable de construire des terrains possédant une structure géomorphologique réaliste, sans faire appel à des simulations d'érosion. En générant d'abord un réseau hydrographique, puis en construisant le relief, nous parvenons à obtenir un terrain respectant le principe d'écoulement des eaux. Pour définir géométriquement chaque cours d'eau, nous nous appuyons sur une classification d'hydromorphologie. L'ensemble de cet algorithme génère le terrain sous la forme d'un arbre de construction définissant une fonction d'élévation.

Enfin, pour conclure, nous réutilisons dans le Chapitre 6, le modèle de comparaison sous forme de diagramme radar (développé et utilisé dans l'état de l'art de cette thèse) pour faire apparaître les qualités et les défauts de nos différentes contributions. Cette analyse permet d'aborder les différentes perspectives liées à nos travaux.

2

État de l'art

Sommaire

C'est par la logique qu'on démontre,
c'est par l'intuition qu'on invente.

— Henri POINCARÉ

Dans ce chapitre, nous présentons un état de l'art dédié à la création de scènes naturelles numériques. Comme il existe un très grand nombre d'articles traitant de ce sujet, il est nécessaire d'exclure certains domaines de notre état de l'art.

Nous nous concentrons sur la notion de terrain (on parlera indifféremment de terrains, de relief ou de topographie). Les notions d'apparence et de décorations ne seront pas évoquées dans cette thèse, bien qu'elles influent grandement sur la perception d'un paysage [1]. Nous ne détaillerons pas non plus les modèles et systèmes utilisés en informatique graphique mais non-spécifiques aux décors naturels (maillages 3D, déformations géométriques, paramétrisation de textures, *etc*). Enfin, nous excluons les techniques de représentation et de visualisation qui ne s'intéressent qu'à manipuler des données d'acquisition provenant de reliefs réels (souvent dans le cadre de la géographie, de la géologie ou de l'astronomie).

Pour résumer : nous cherchons à étudier à la fois **les représentations** mais aussi **les méthodes d'édition** et **les algorithmes de génération** qui s'intéressent aux **reliefs** et qui soient **utilisables dans le contexte d'une production artistique**

Pour l'ensemble des travaux présentés dans la suite de chapitre, nous citons un maximum de ressources dans un objectif de pédagogie [2]et d'exhaustivité.

Dans la suite de ce chapitre, nous présentons un nouveau modèle de comparaison (Section 2.1) qui évalue les représentations ou les méthodes d'édition ou de génération selon 6 critères. Cette classification sera utilisée tout au long de cet état de l'art pour analyser les différentes familles d'algorithmes. Nous avons décidé de structurer les travaux issus de la littérature selon deux grandes catégories : les modèles de représentations (Section 2.2) et les algorithmes de manipulation de terrains (Section 2.3). Une conclusion générale (Section 2.4) complète cet état de l'art en rappelant les limites courantes du domaine et en présentant certaines pistes intéressantes qui sont actuellement étudiées.

2. La visualisation correcte et efficace de l'ensemble des objets et phénomènes naturels est un problème difficile qui combine modélisation et rendu ; Neyret en expose les principales méthodes de résolutions dans [261] et [262].

2. Afin de comprendre l'historique d'une idée et d'avoir un maximum d'explications nous citons (quand ils sont disponibles) actes de conférences nationales, rapports de recherche, posters, conférences internationales, mémoires de Master ou de thèse, livres, *etc*.

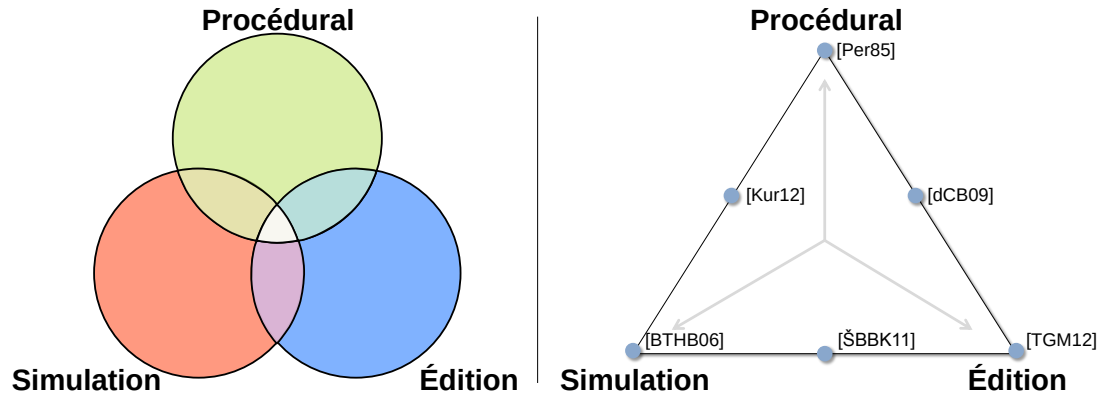


FIGURE 5 : Ces classifications, utilisées par PEYTAVIE et MARÉCHAL, permettent de regrouper les algorithmes de modélisation et de génération de terrains en trois familles de méthodes. Quelques articles ([34, 96, 194, 288, 365, 372]) sont placés sur le triangle de droite.

2

2.1 Classification des modèles de terrains

Il n'existe que peu de travaux de classifications des méthodes de représentation de terrains. Concevoir une nouvelle méthodologie de comparaison est un acte subjectif puisqu'on propose des critères qui nous paraissent importants (et à l'inverse on en ignore certains). Si quelques-uns de ces critères de comparaisons sont mesurables (temps d'exécution, taille mémoire), d'autres sont plus difficile à quantifier (l'outil est-il meilleur ? plus réaliste ? plus intuitif ?).

De la même manière, comparer des reliefs peut se faire selon des critères mesurables (l'histogramme des altitudes, des pentes, *etc*) ou non (*i. e.* le réalisme ou la beauté visuelle d'un terrain). Ces critères non quantifiables qui définissent généralement des notes n'ont été que peu étudiés. Une autre approche consiste à se baser sur des études perceptuelles de groupes (*user studies*) et de laisser les sujets décider quel terrain leur paraît réaliste ou beau [3]. Nous pensons qu'élaborer de nouveaux moyens de mesurer quantitativement le réalisme d'une scène virtuelle est un problème ouvert. De la même manière, il serait utile de concevoir un protocole formalisé de *user study* centré sur les questions de perception du réalisme d'un terrain.

Précédentes taxonomies

PEYTAVIE [297] et MARÉCHAL [226] ont proposé dans leurs thèses de classer les méthodes de génération et d'édition d'objets selon trois familles : **les méthodes procédurales**, **les simulations** et **les méthodes d'édition**. Originellement, cette classification est représentée sous la forme d'un diagramme de Venn (Fig. 5 gauche). Par la suite, c'est un triangle défini par trois caractéristiques (aspect procédural, simulations, édition) qui est utilisé pour placer et différencier les différentes méthodes entre elles [131] (Fig. 5 droite). Cette classification est intéressante mais certaines mé-

3. Des travaux proches ont par exemple été réalisés pour connaître l'importance de certains phénomènes optiques pour la visualisation d'océans animés [49].

thodes combinent à la fois des simulations et des techniques procédurales. De plus, il n'y a aucune notion de qualité de résultats [4].

Un état de l'art publié récemment par NATALI *et al.* [257] s'est intéressé à classifier les techniques de représentation de terrains. Ils proposent deux taxonomies. La première regroupe les représentations de terrains en **trois familles suivant la quantité de données** qu'elles manipulent (*data-free*, *sparse-data*, *dense-data*). La deuxième classification repose sur **la manière dont l'infographiste crée et édite un relief** (*workflow taxonomy*). Ces deux taxonomies ont de nombreux défauts : on ne voit pas les avantages intrinsèques de chaque représentation ou mode d'édition. Il est difficile de quantifier le contrôle ou même le coût-mémoire.

Un autre état de l'art sur les représentations procédurales a été proposé par SMELIK *et al.* [348]. Dans cet article, les auteurs cherchent à référencer l'ensemble des travaux en modélisation procédurale en informatique graphique, pour la génération de mondes virtuels. Ils **classifient ces méthodes selon le type d'objets à générer** (les terrains, la végétation, les cours d'eau, les routes, les villes, les bâtiments et les intérieurs) et selon **les familles d'outils utilisés** (purement stochastiques, avec de l'IA, avec des simulations, avec des grammaires, avec des données, ou avec des outils géométriques). Pour l'utilisation de **méthodes et de représentations procédurales utilisables pour les jeux-vidéos**, nous invitons le lecteur à lire les états de l'art écrits par DE CARLI *et al.* [93] et HENDRIKX *et al.* [159].

Concernant la théorie des méthodes procédurales (*search-based Procedural Content Generation*) [5], il existe **une taxonomie plus formelle** proposée par TOGELIUS *et al.* [380]. Les auteurs proposent 5 critères pour caractériser une technique procédurale (en-ligne/hors-ligne, nécessaire/optionnel, aléatoire/paramétré, déterministe/stochastique, correct par construction/test).

Aucune de ces taxonomies ne nous convient. Nous proposons donc un modèle selon 6 critères, et représenté par un diagramme radar [6]. Nous proposons d'étudier à la fois les modèles de représentations et les outils d'édition de terrains selon les mêmes critères de comparaison.

Notre modèle de comparaison

Nous définissons six critères pour caractériser les modèles et les algorithmes qui sont :
- **[M] l'utilisation Mémoire**
- **[V] la Vitesse de calcul**.
- **[I] l'Intuitivité**
- **[X] l'eXpressivité**
- **[R] le Réalisme**
- **[E] les Échelles de taille et résolution**

Nous notons chaque critère sur une échelle entre 1 et 4. Dans le Tableau 1, nous donnons quelques points de repères pour chaque critère.

4. Certaines simulations peuvent être décevantes d'un point de vue réalisme et inefficaces alors qu'une méthode de modélisation interactive arrivera à créer plus rapidement et plus facilement une scène complexe.
5. une vision plus proche de « l'intelligence artificielle » (au sens large) que de l'informatique graphique
6. Ces diagrammes sont généralement considérés comme dangereux car leur lecture dépend de la position des différents critères. Dans notre cas, nous exploitons les adjacences.

Tableau 1 : Les descriptions des notes sont indicatives. L'échelle est donnée sous la forme d'une plage de résolution à une dimension mais peut s'étendre en 2D/3D. Pour la taille mémoire, les algorithmes traitant des données (simulations, agents sur grille) seront notés en fonction de la taille des structures sur lesquels ils peuvent s'exécuter. Pour la vitesse, les modèles de représentation seront notés en fonction de leur facilité à être visualisés (rasterisation ou lancer de rayons).

	Critères	Notes			
		★	★ ★	★ ★ ★	★ ★ ★ ★
●●	**Échelles**	m—10 m	m—km	cm—10 km	∞ de résolutions
●	**Mémoire**	≈ 100 Go	≈ 10 Go	≈ 10 Mo	≈ 5 ko
●●	**Vitesse**	≈ 3 min	≈ 10 s	≈ 1 s	≈ 50 ms
●	**Intuitivité**	pas d'outils	généralistes, détournés	adaptés, spécialisés	simples, intuitifs, prédictibles
●●	**eXpressivité**	juste relief planaire	certains phénomènes géométriques	tous phénomènes géométriques (surplombs, volumiques, autorise simu. érosions)	géométrique + autres interactions (lumières, liquides, *etc*)
●	**Réalisme**	aucune garantie	pas d'artefact visuel/géométrique	possible d'améliorer manuellement le réalisme	facile de garantir l'hydrologie

Le diagramme de Venn (puis le schéma triangulaire) introduit par PEYTAVIE et MARÉCHAL, proposait trois familles de méthodes : procédurales, interactives et simulations. On qualifie traditionnellement d'efficaces les méthodes procédurales, de réalistes les simulations et de contrôlables les méthodes interactives (bien que certaines simulations soient plus efficaces que d'autres, ou que certaines méthodes interactives utilisent des principes de la simulation, *etc*).

Il est possible d'**adapter nos six critères selon ces trois axes qualitatifs : efficacité, qualité et contrôle** (Fig. 6). Les méthodes efficaces (**[EMV]**) sont celles qui sont légères en mémoire, rapides et adaptatives. Les outils les plus contrôlables (**[VIX]**) sont interactifs, intuitifs, et permettent d'exprimer toutes les intentions de l'artiste. Les représentations et algorithmes qui cherchent avant tout la qualité visuelle (**[REX]**) s'intéressent à construire des terrains réalistes, avec un mélange d'éléments de plusieurs échelles et de n'importe quels forme ou type.

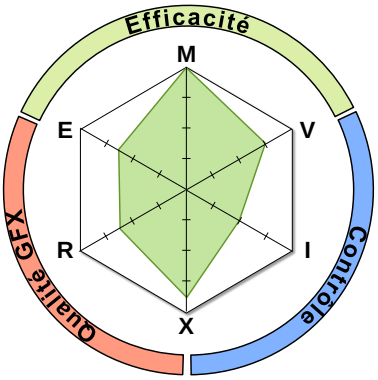


FIGURE 6 : Notre classification.

Nous pensons que ce schéma hexagonal et sa taxonomie associée est **plus précis** que les classifications existantes et **permet de mieux différencier** les différentes méthodes. De plus, s'inscrivant dans la continuité des travaux de PEYTAVIE et MARÉCHAL, il propose à l'utilisateur une lecture de plus haut niveau selon trois axes principaux.

2.2 Représentations de terrains

Un objet 3D peut être représenté de deux façons : avec des données qui décrivent l'objet sans calcul, ou bien avec une procédure (qui peut s'appuyer sur des données partielles). On parlera dans le premier cas de **représentations explicites** et dans le deuxième cas de **représentations procédurales**. Cette frontière est floue car certains modèles explicites s'appuient sur des données incomplètes (*e. g.* lorsqu'on manipule un nuage de points) et certains modèles procéduraux s'appuient sur des points de contrôles qui feront partie de la surface (dans le cas des interpolations, du krigeage ou d'esquisses de lignes de crêtes).

En informatique graphique, une définition classique propose de séparer les modèles 3D en deux catégories : les **représentations surfaciques** et **représentations volumiques** (Fig. 7). Les premières représentent uniquement l'enveloppe d'un objet. Elles sont plus efficaces à visualiser et à stocker et de nombreux logiciels peuvent les manipuler. Formellement, ce qui distingue une représentation volumique d'une représentation surfacique, c'est la possibilité d'avoir une procédure qui puisse déterminer si un point de l'espace et à l'intérieur ou à l'extérieur de l'objet. Par conséquent, les modèles surfaciques permettent de représenter des objets non constructibles (comme des bouteilles de Klein). En pratique, les modèles volumiques permettent également de représenter plus facilement la composition des objets ce qui permet de manipuler la notion de densité qu'on retrouve dans les gaz ou les liquides. Dans le cadre réel, un terrain est une structure géologique solide qui est composée de plusieurs éléments hétérogènes : il faudrait donc utiliser des représentations volumiques. Mais le sol n'étant pas transparent, et les compositions

rocheuses hétérogènes étant difficiles à manipuler et à visualiser, on reconstruit généralement plusieurs modèles surfaciques.

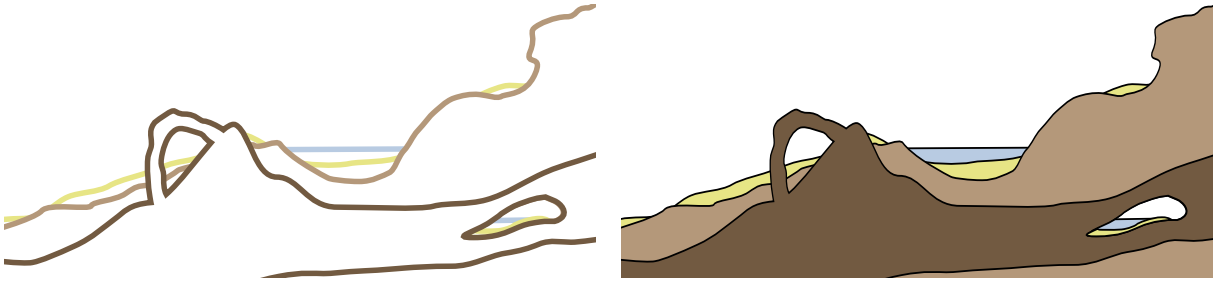


FIGURE 7 : (Gauche) Les représentations surfaciques ne représentent que l'enveloppe des objets. (Droite) Avec les représentations volumiques, il est facilement possible de savoir si un point est à l'intérieur ou à l'extérieur d'un objet ; de façon formelle, les surfaces orientées fermées (*i. e.* sans bord) définissent des objets volumiques.

Une vraie distinction qui nous importe concerne les structures de données aptes à représenter des terrains **non-planaires (3D)** ou uniquement **planaires (2,5D)**. En effet, certains modèles de représentation sont plus efficaces que d'autres en termes de mémoire ou d'accès en lecture et écriture mais ils ont une limite importante : ces modèles stockent un relief encodé sous la forme d'une fonction 2D.

Cette distinction limite fortement l'expressivité de ces représentations. Les modèles non planaires peuvent être mélangés à des représentations planaires (l'intégration de modèles 3D classiques sur des cartes de hauteurs se fait couramment dans les moteurs de jeux) mais il résulte généralement des artefacts visuels (*i. e.* maillages non correctement raccordés). De plus, le fait d'utiliser plusieurs modèles de représentation limite fortement les opérations de modifications que l'on peut effectuer (simulation, édition).

2.2.1 Acquisition de données réelles (géologiques ou géographiques)

Dans le cadre de la géographie et de la géologie (notamment dans l'industrie pétrolière ou minière), les chercheurs s'intéressent à reconstruire des terrains à partir de données réelles. Ces données peuvent être surfaciques ou volumiques (Fig. 8). Les données surfaciques sont principalement captées par des scanners mobiles (type LIDAR) ou par satellites. Les données volumiques sont, elles, capturées en propageant des ondes sonores dans le sous-sol et en capturant leurs échos avec des géophones. L'analyse de la vitesse, de l'amplitude et de l'angle de retour de l'écho permet de détecter la composition du sol.

Ces données peuvent être très volumineuses et il faut souvent des algorithmes spécialisés pour les traiter et les visualiser. Les cartes de hauteurs de la Terre (d'une résolution de 1 point tous les 100m) font plusieurs centaines de Go et des techniques de *streaming* qui permettent de charger ces données petit à petit sont nécessaires lors de la visualisation. Les relevés soniques étant de nature volumique, ils ont un impact mémoire d'un ordre de complexité supérieur et

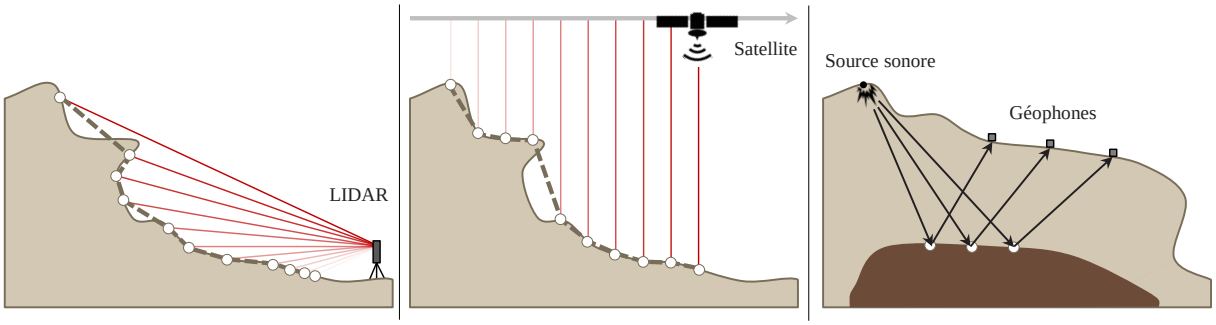
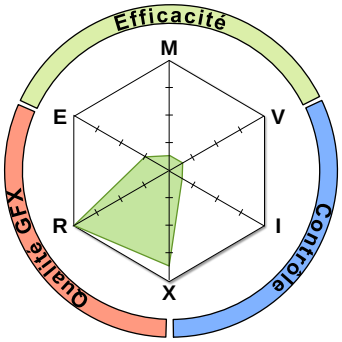


FIGURE 8 : Quelle que soit la nature des données captées – nature ponctuelle (LIDAR), surfacique (images satellites) ou volumique (relevés soniques) –, il est nécessaire de reconstruire des modèles 3D visualisables à l'aide d'algorithmes coûteux en temps de calcul.

prennent souvent plusieurs To d'espace de stockage. De plus, ces données ne sont pas visualisables directement en tant que telles et il est alors nécessaire de reconstruire des modèles 3D géologiques complexes.

Analyse : Les données réelles provenant de nuages de points capturés à la surface des reliefs ou de relevés géologiques permettent de reconstruire très finement des terrains réels, y compris au niveau de leur composition interne. **Leur coût-mémoire est souvent exorbitant** et il est nécessaire de traiter lourdement ces données pour pouvoir les visualiser. De plus ces données sont **peu intuitives** (il est souvent nécessaire de faire des retouches manuelles des modèles produits) et, sans traitement, ne sont que difficilement utilisables dans le cadre pédagogique.



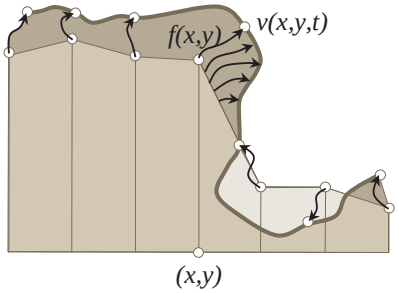
2.2.2 Représentations explicites planaires (2,5D)

Les représentations planaires encodent le terrain sous la forme d'une fonction 2D qui représente une surface surélevée :

$$f : \mathbf{R}^2 \to \mathbf{R}$$
$$(x,y) \mapsto z = f(x,y).$$

Cela implique qu'il n'y a qu'une élévation possible en tout point du plan, rendant impossible la représentation de surplombs ou de grottes. Cette limitation est généralement contournée en intégrant des modèles 3D classiques supplémentaires dans la scène (on manipule alors deux formalismes en parallèle).

Une solution élégante pour transformer un modèle 2,5D en modèle réellement 3D est proposée par GAMITO et MUSGRAVE [138]. En déformant la surface du relief par un opérateur d'advection (la surface est déformée continuellement le long d'un champ de vecteurs), ils garantissent que la nouvelle surface sera à la fois 3D, simplement connectée (topologiquement homomorphe à un plan) et ne contiendra aucune auto-intersection.

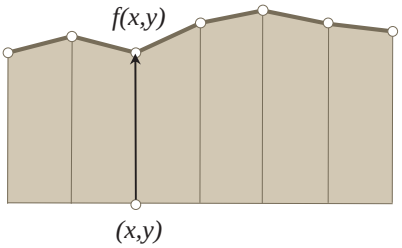


Les géographes utilisent presque exclusivement des modèles surfaciques. De nombreux livres et journaux scientifiques utilisent ces représentations. Pour une description générale des différents modèles de représentation des DEM (*digital elevation model*) planaires, nous conseillons au lecteur le chapitre 9 de *Geographical information systems : Principles and Technical Issues* [170] écrit par HUTCHINSON et GALLANT. Deux livres complémentaires et plus récents sont *Digital Terrain Modeling : Principles and Methodology* [218] écrit par LI et QING en 2005 et *Abstracting Geographic Information in a Data Rich World* [151] écrit par GUILBERT *et al.* datant de 2014.

Quelques modèles planaires spécialisés ou expérimentaux ont été proposés dans des articles et ne sont pas (ou peu) utilisés par l'industrie. ATLAN et GARLAND [12] proposent par exemple de représenter les terrains sous forme de *quadtrees* d'ondelettes. Cette structure est rapide et relativement légère en mémoire et permet d'éditer et de visualiser de grands terrains. PATEL [279] propose un modèle de génération basé sur des polygones correspondant aux cellules de Voronoï d'un nuage de points. Sa structure manipule à la fois le diagramme de Voronoï et son dual, la triangulation de Delaunay pour construire des îles avec leur réseau hydrographique.

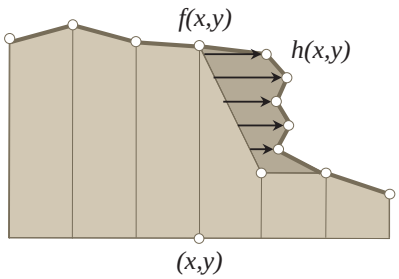
2.2.2.1 Cartes de hauteurs et structures régulières

Le modèle de représentation du terrain le plus classique est la grille régulière de hauteurs (*heightmap* ou encore *heightfield*) [7] (Fig. 9 gauche). Une des limitations des cartes de hauteurs est leur incapacité à représenter des surplombs et des galeries souterraines : à toute cellule de la grille correspond une unique élévation.



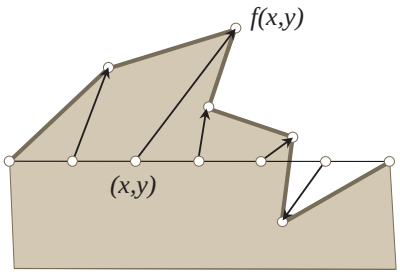
Il est important de savoir précisément quelle position géométrique correspond à quelle élévation stockée dans l'image. Trois interprétations co-existent : soit chaque cellule définit l'élévation au centre de celle-ci, soit chaque cellule correspond à l'élévation du point de la grille duale, soit on combine les deux interprétations. Les cartes de hauteurs étant des textures comme les autres, elles sont gérées par les GPU de manière efficace. Ainsi, il est possible de filtrer efficacement les images rasters. Les cartes de hauteurs peuvent également être utilisées comme des textures décrivant les détails de déformation d'une surface (*bump-mapping*, *etc*).

7. On stocke généralement ces cartes de hauteurs sous forme de textures 2D en niveaux de gris



Il est possible d'utiliser une carte de hauteurs pour déformer un terrain (ou un objet) selon un plan (qui n'est pas Oxy). Dans le cas des terrains, c'est souvent un déplacement horizontal qui permet d'obtenir des reliefs avec des surplombs. Les objets 3D peuvent également être déformés selon un plan local tangent à la surface. Cette déformation peut être géométrique ou seulement simulée au niveau du calcul de la lumière (comme dans le cas du *bump-mapping*).

Au lieu de représenter une élévation, il est possible de stocker directement un champ de vecteurs de déformation (généré avec quelques contraintes pour garantir qu'il n'y ait pas d'auto-intersection du maillage). Une telle structure appelée *vector fields terrains* [234] a été proposée par BONFIRE STUDIOS et MICROSOFT ; elle est intégrée dans le jeu-vidéo HALO WARS et permet d'obtenir des surplombs facilement sans devoir intégrer des modèles 3D supplémentaires. Par contre, cette structure nécessite 3 fois plus de données à stocker en mémoire.



Il est possible de transformer une carte de hauteurs en un ensemble de triangles pour les rasteriser. La topologie de la grille est codée de manière implicite : chaque pixel est relié à ses 4 voisins (ce qui correspond à des sommets du maillage ayant soit une valence de 4 quand on utilise des quadrangles, soit une valence de 6 quand on utilise une subdivision régulière en triangles). Le maillage est construit en générant une suite de bandes de triangles.

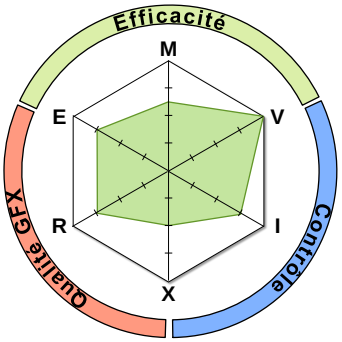
FIGURE 9 : Les structures régulières (cartes de hauteurs, grilles d'hexagones) sont des modèles classiques pour représenter des terrains. Efficaces, ces modèles peuvent même être visualisés à l'aide de techniques de lancer de rayons en temps-réel.

Un autre moyen de visualiser ces cartes de hauteurs est d'utiliser des algorithmes de lancers de rayons. Les cartes de hauteurs représentant des surfaces planaires (sans surplombs), il est possible d'utiliser cette propriété pour optimiser les calculs de rendu. Certains moteurs de jeux[8] [9, 267, 268] rendent le terrain directement par lancer de rayons (Fig. 9 centre).

8. Ce type de moteur est appelé historiquement *voxel renderer* même s'il s'agissait uniquement de cartes de hauteurs (ce terme inexact est maintenant mêlé à des moteurs de rendu 3D par rasterisation où l'on manipule des voxels comme MINECRAFT [246]).

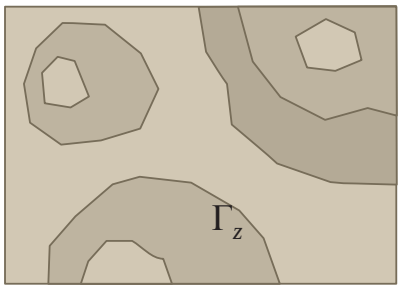
McGuire et Sibley [235, 236] montrent qu'il est possible de déformer la grille régulière pour obtenir des triangles plus équilibrés. Cette transformation améliore la qualité de rendu des terrains (au niveau du calcul des normales par exemple).

Une autre structure régulière souvent utilisée est la grille d'hexagones (Fig. 9 droite). Ces derniers ont la propriété de partager une arête avec chacun de leur voisin. Leur structure est donc plus régulière et approxime bien mieux la distance Euclidienne. C'est une des raisons pour laquelle cette structure est très utilisée dans les jeux-vidéos et les jeux de plateaux. Plus difficile à manipuler, elle permet tout de même de représenter des reliefs 2,5D. Pour une étude complète des différents systèmes de coordonnées et des algorithmes utiles pour manipuler une grille hexagonale, nous laissons le lecteur se tourner vers les travaux de Patel [281]. Dixon *et al.* [99] adaptent les algorithmes de subdivision classiques pour fonctionner sur plusieurs structures régulières (triangles, quads, hexagones) ; ils proposent également une structure mixte à base de polygones qui permet d'utiliser plusieurs formes en même temps.



Analyse : les cartes de hauteurs sont **la représentation surfacique la plus classique** des terrains. Très étudiées, elles nécessitent peu de mémoire et sont très rapides à visualiser aussi bien par rasterisation que par lancer de rayons. Il est possible d'utiliser des outils classiques de peinture pour les éditer. La grille est également la structure de données utilisée pour les analyses de flots et les simulations Eulériennes. Ces modèles sont relativement adaptés aux variations d'échelles (grâce à des systèmes de *streaming* et de tuiles ou à des structures accélératrices hiérarchiques). Malgré tout, ces représentations ne permettent pas du tout de gérer des surfaces non-planaires, ni des matériaux volumiques hétérogènes.

2.2.2.2 Cartes d'isocontours



Les terrains planaires peuvent être stockés sous forme d'isocontours (Fig. 10). Un isocontour est une ligne polygonale ou une courbe qui va représenter un même niveau d'élévation. On parle également de courbe de niveau ou isoplèthe d'altitude. Chaque ligne de niveau est obligatoirement encadrée par deux autres lignes et il n'est pas possible qu'une ligne de contour intersecte une autre voisine. Ces contours sont généralement présents sur les cartes topographiques papiers afin d'indiquer la forme des reliefs sur un support plan.

Le stockage de modèle 3D de terrains sous forme d'ensemble d'isocontours présente l'avantage d'avoir des informations vectorielles (les courbes de niveaux) qui sont relativement légères en mémoire. Ces cartes d'isocontours sont souvent utilisées dans le cadre de la géographie. Les structures informatiques sont généralement construites à partir de relevés numériques (nuages

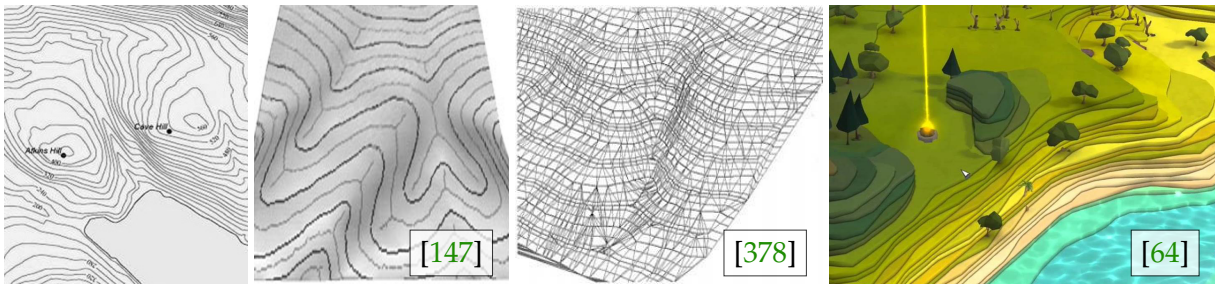
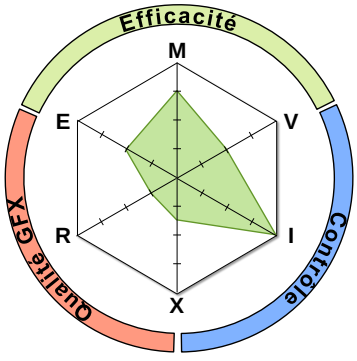


FIGURE 10 : Les iso-contours sont très utilisés en géographie car ces structures permettent de décrire de façon intuitive les formes des reliefs. Pour visualiser le relief, il est nécessaire de reconstruire une triangulation sans artefacts visuels.

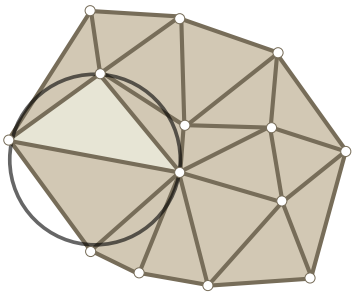
de points, cartes de hauteurs) bien qu'il soit possible de concevoir des algorithmes capables de scanner et de reconstruire numériquement des cartes sur support papier [322].

Pour être visualisées, les cartes d'isocontours doivent être transformées en des maillages 3D. L'utilisation de techniques de reconstruction de maillages classiques entraîne la création d'artefacts visuels. En particulier, des zones composées de triangles plats peuvent apparaître (généralement dans des terrains à fort relief et modélisés avec un nombre restreint d'isocontours différents). Ces problèmes sont généralement corrigés en extrapolant de nouveaux points sur les courbes de niveaux (à l'aide d'une étude du diagramme de Voronoï des échantillons [378], en cherchant à appareiller au mieux les lignes de contour deux à deux [343, 344] ou en analysant les points extrema du relief [416]). L'élimination d'artefacts visuels (triangles plats) peut passer par l'augmentation du nombre d'échantillons [147] ou en cherchant à construire un terrain C^1 [164]. On peut également chercher à améliorer la qualité des bandes de triangles entre deux isocontours en particulier lors des embranchements topologiques [420]. Enfin, des travaux s'intéressent à optimiser les temps de calcul en s'aidant du GPU [422].

Analyse : moins polyvalentes que les structures régulières, les cartes d'isocontours ont l'avantage d'être **intuitives à manipuler** sans nécessairement reconstruire le terrain. Mais ces modèles, relativement légers en mémoire, doivent être transformés en maillages 3D pour être visualisés ce qui a un coût non négligeable en terme de calculs. De plus, les modèles reconstruits peuvent présenter des artefacts visuels. Un autre défaut des cartes d'isocontours est le fait qu'elles ne sont pas du tout adaptées pour faire des calculs de simulations (il faut, là encore, passer par d'autres modèles de représentations). Enfin, ces structures sont planaires et ne peuvent pas représenter des surplombs.



2.2.2.3 TINs



Les TIN (*Triangulated Irregular Networks*) sont des maillages surfaciques planaires (Fig. 11). Inventés parallèlement par PEUCKER *et al.* [294, 295] et GOLD *et al.* [148], ils sont extrêmement utilisés dans le cadre de logiciels de systèmes d'information géographique (GIS ou *Geographic Information Systems*). Ils sont généralement construits par triangulation de Delaunay sur un ensemble de points caractéristiques de la surface du terrain. Les TINs utilisent un nombre plus important de triangles dans les zones à détailler (généralement où l'altitude varie de façon importante) tout en représentant de manière plus grossière les zones plus plates.

Grâce à leur propriété planaire, il est possible d'optimiser les calculs de visualisation en incorporant des mécanismes de niveaux de détails. De la même manière, les simulations physiques (écoulements d'eau, de vents) sont fortement simplifiées sur des maillages triangulaires formant une surface planaire.

Contrairement aux cartes de hauteurs, il est nécessaire de stocker la connectivité des triangles (ou alors de la calculer par un algorithme coûteux) ; cela complique la mise en place de solutions de *streaming*. À l'inverse, les maillages sont souvent de meilleure qualité grâce à des techniques de raffinement car ils permettent d'éviter les problèmes de crénelage qui proviennent de l'utilisation d'une grille régulière.

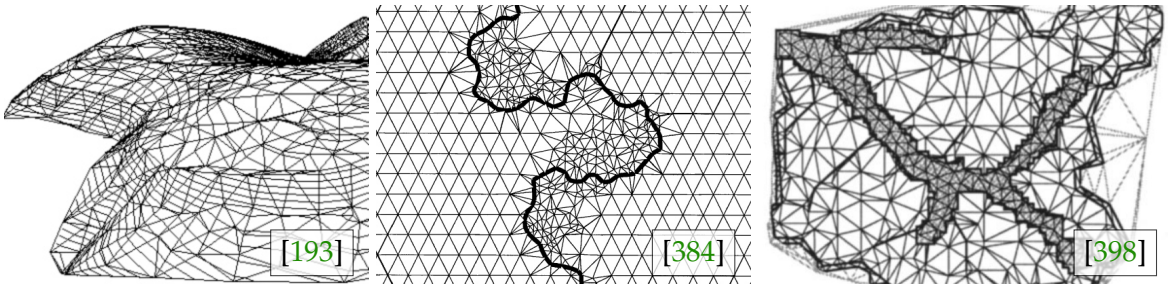
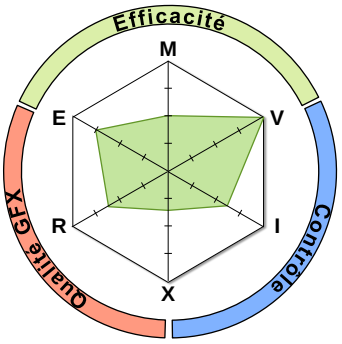


FIGURE 11 : Les *Triangulated Irregular Networks* sont plus complexes à manipuler que les cartes de hauteurs mais permettent d'obtenir des maillages plus propres (en particulier sur les forts reliefs qui ne suivent pas les axes d'une grille comme les bords de rivières ou de routes).

Analyse : ces structures sont très rapides à visualiser, les modèles triangulaires surfaciques étant les données **les plus adaptées aux GPU modernes**, et ont un coût-mémoire acceptable (mais contrairement aux autres structures planaires, il est nécessaire de stocker la connectivité des triangles). Les TINs permettent d'utiliser des algorithmes de visualisation plus avancés comme des mécanismes de niveaux de détails. Ces structures ne sont, par définition, absolument pas adaptées à la représentation de surplombs ou d'objets volumiques.



2.2.3 Représentations explicites non-planaires (3D)

Les représentations non-planaires peuvent permettre de modéliser des terrains 3D avec des surplombs ou des grottes. Ces modèles sont plus expressifs, mais moins performants que les représentations planaires. La plupart des outils de modélisations classiques ont été conçus pour manipuler des structures non-planaires. À l'inverse, ces modèles sont peu utilisés lors des simulations d'érosion ; en effet, le coût en terme de calculs est souvent d'un ordre de complexité plus grand que le même type de calcul avec des surfaces planaires.

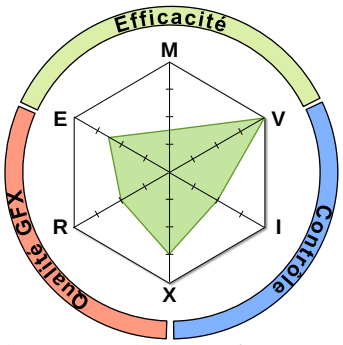
Dans l'industrie, les moteurs gèrent souvent à la fois des représentations planaires et des représentations avec surplombs. Dans le cadre d'un jeu-vidéo par exemple, on ne vérifie pas forcément que les maillages se raccordent proprement (sans intersection ni jonction en T). Il existe pourtant dans la littérature quelques structures mixtes 2.5D/3D qui mêlent cartes de hauteurs et voxels comme celle proposée par KOCA et GÜDÜKBAY [190].

2.2.3.1 Maillage surfacique 3D

Un maillage est un ensemble de points, d'arêtes et de faces qui définissent la surface d'un objet (Fig. 12). Les faces correspondent généralement à des triangles bien que des quadrilatères ou des polygones convexes puissent être utilisés. Un terrain n'étant qu'un objet surfacique particulier, il est possible d'appliquer les mêmes algorithmes et les mêmes modèles de représentation que pour n'importe quel autre maillage. Les représentations classiques de maillages sont : par faces-sommets, par arêtes ailées, par demi-arêtes ou par quad-arêtes [77].

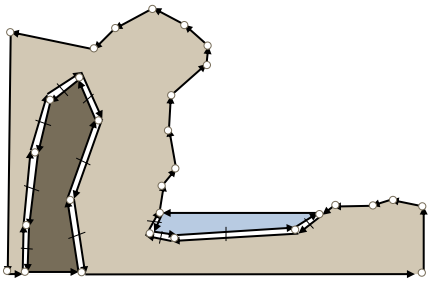
Utiliser des structures de maillages surfaciques classiques présente deux avantages majeures. Tout d'abord, le moteur de rendu n'aura qu'une structure à manipuler et cette gestion unifiée permet de faciliter le développement ou la maintenance du moteur. L'autre avantage réside dans l'immense éventail d'outils à disposition des infographistes pour créer et éditer des modèles 3D. Bien entendu, l'ensemble de ces outils n'est pas spécialement conçu pour manipuler des reliefs de terrains.

FIGURE 12 : Souvent, les maillages 3D classiques sont utilisés pour représenter des objets dans le relief du terrain. Ces structures non-spécialisées coûtent plus cher en mémoire et ne sont pas forcément adaptées aux différentes simulations de phénomènes naturels.



Analyse : ces structures sont **très rapides à visualiser** (les modèles triangulaires surfaciques étant les données les plus adaptées aux GPU modernes), ont un coût-mémoire moyen (bien que des solutions de *streaming* ou de niveaux de détails existent), utilisent des maillages adaptatifs pour gérer plusieurs objets de résolutions hétérogènes et permettent de représenter tous types de terrains surfaciques (mais plus difficilement les liens topologiques entre ces objets ou avec des volumes). Il est possible d'adapter des algorithmes de simulation d'écoulement d'eau ou de vent sur ces structures même si elles ne sont pas adaptées au départ. L'édition de ces modèles se fait généralement avec des outils génériques de géométrie.

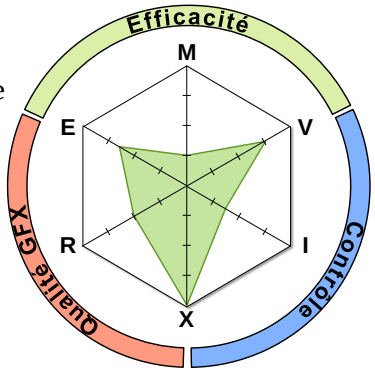
2.2.3.2 Cartes combinatoires et cartes généralisées



D'autres structures de données, plus efficaces quand il s'agit d'appliquer des opérations qui nécessitent de connaître la topologie [92] peuvent être également utilisées pour représenter des terrains. Les cartes combinatoires peuvent représenter des volumes 3D et stocker l'information que deux volumes sont incidents. Elles reposent sur une structure de brin qui unifie la représentation : ainsi, un sommet, une arête, une face ou un volume ne sont que des ensembles différents de brins. Depuis un brin, il existe des procédures pour d'obtenir facilement l'arête, la face ou le sommet correspondants. De la même manière, il est possible de construire des procédures plus complexes pour obtenir les sommets voisins à un sommet, ou les faces connexes à une face, *etc*.

Les cartes généralisées sont une extension de ces travaux qui, au prix d'un doublement du nombre de brins (4 pour représenter une arête au lieu de deux) permettent de construire des objets volumiques non orientables comme des bouteilles de Klein.

Des travaux pour l'application de ces structures pour représenter des terrains volumiques existent, qu'ils soient à destination des géologues [56, 212] ou des infographistes [37, 39, 40, 41, 296] (Fig. 13).



Analyse : ces structures sont plus spécialisées que les maillages traditionnels et permettent de **facilement manipuler la topologie**. Elles peuvent représenter plus de phénomènes (comme des objets volumiques) et permettent de faire des calculs de simulation plus facilement. À l'inverse, ces structures sont moins simples à manipuler, moins rapides et beaucoup plus lourdes en mémoire. Récemment, elles ont souvent été utilisées à la frontière entre l'informatique graphique et les géosciences. De plus, on manipule souvent les cartes combinatoires lors de l'édition, puis on extrait un maillage optimisé pour pouvoir visualiser efficacement le relief.

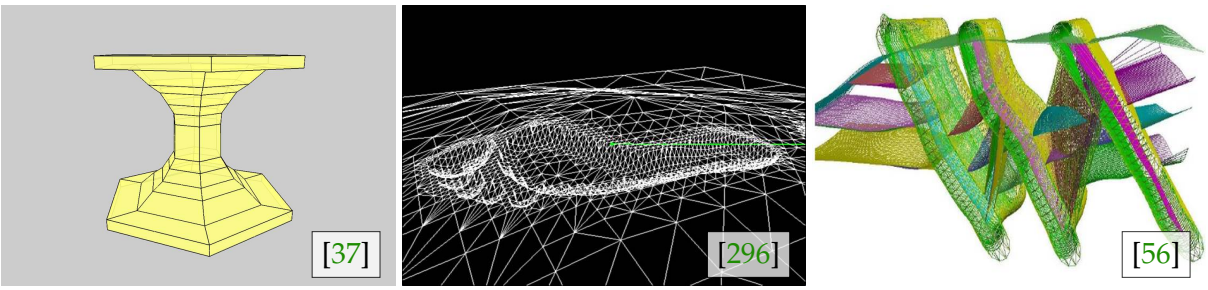
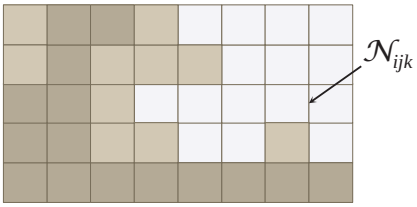


FIGURE 13 : Les cartes combinatoires et les cartes généralisées sont des structures permettant d'encoder facilement les relations de voisinage entre deux surfaces ou entre deux volumes. Elles sont beaucoup plus coûteuses en mémoire que de simples maillages.

2.2.3.3 Voxels et piles de matières



Pour représenter des structures complexes, il est possible d'utiliser une représentation discrète et volumique. La plus simple et la plus utilisée de ces représentations est le voxel (pour *volumic element*). Chaque objet est composé d'un ensemble de cubes dans l'espace, placés dans une grille régulière 3D (Fig. 14 gauche). Avec une résolution suffisamment fine, ces cubes peuvent donner une bonne approximation de la surface des objets que l'on modélise.

La plupart des logiciels d'édition de voxels cultivent une ressemblance avec les outils de sculpture 3D (comme ZBRUSH [302]). Plusieurs moteurs de jeux récents s'appuient sur une représentation des objets à base de voxels (MINECRAFT [246], EVERQUEST NEXT [76]) qui permet à tout utilisateur de modifier intuitivement l'environnement (ajout de matière, creusement).

Pour stocker ces voxels, la méthode la plus simple est d'utiliser une texture 3D. Ces images sont très volumineuses et il est souvent utile d'essayer de les compresser au mieux (par exemple avec un encodage RLE traditionnel). Les SVO ou *Sparse Voxel Octree* [201, 202, 203] sont un moyen de stocker encore plus efficacement des scènes géométriques en minimisant la résolution de la grille voxelique dans les zones où cela est possible (Fig. 14 centre). En 2013, KAMPE *et al.* [183] proposent d'améliorer cette représentation hiérarchique à l'aide d'un graphe acyclique orienté qui permet d'augmenter l'efficacité mémoire de plusieurs ordres de magnitude. Si les SVO encodent efficacement les régions vides de l'espace, utiliser un DAG revient à encoder efficacement les régions identiques de l'espace (Fig. 14 droite).

Des structures alternatives aux voxels binaires (présence ou absence de matière) et spécialement étudiées pour la représentation des terrains ont été proposées au cours des dernières années (Fig. 15). BENEŠ et FORSBACH proposent [30] d'utiliser une grille d'empilement de matières (*layered materials*) qui représentent la superposition de plusieurs couches géologiques (roche, sédimentaire, eau). Ce modèle ne permet pas de représenter des surplombs mais est tout de même très utile lors des simulations d'écoulement d'eau ou d'instabilité gravitaire.

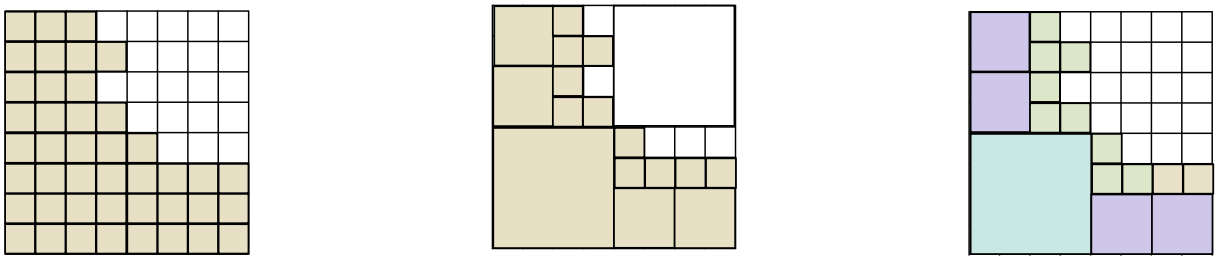
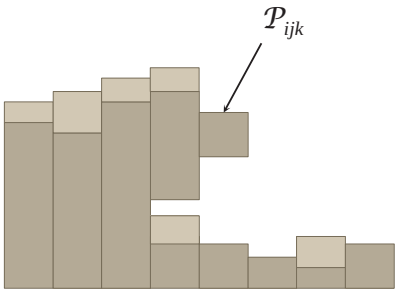
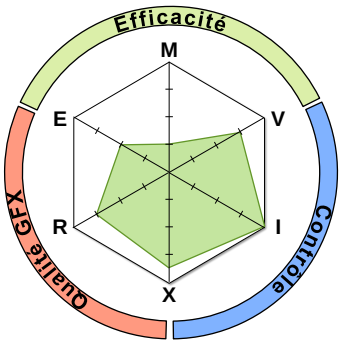


FIGURE 14 : Les SVO permettent d'encoder de façon très efficaces les voxels, en particulier les régions homogènes (complètement vides ou pleines) de l'espace. L'utilisation d'un DAG permet également d'optimiser les sous-régions identiques d'une scène.



Poursuivant quelques travaux préliminaires [172], PEYTAVIE *et al.* [297, 298, 299] étendent cette représentation pour modéliser des reliefs complexes comprenant des surplombs et des grottes en proposant d'utiliser des piles de matières (dont l'air). Ces piles sont plus complexes à gérer mais plusieurs algorithmes d'érosion et d'édition locale ont été proposés pour facilement ou automatiquement sculpter des arches ou des cavernes. Cette représentation contrôlable et intuitive du terrain est par la suite transformée en une surface lisse calculée par convolution. Une autre représentation similaire est proposée par SANTAMARÍA-IBIRIKA *et al.* [335, 336] qui utilisent aussi une structure de piles de matières mais dont les densités sont mixtes. Ils couplent ce modèle à des algorithmes de générations qui peuvent construire des terrains volumiques avec des couches géologiques, des poches de minerais et des caves.

FIGURE 15 : Il est très intuitif de construire ou d'éditer des scènes représentées avec des voxels ou leurs structures dérivées (piles de matières, SVO). Si ces modèles sont très lourds en mémoire, ils permettent de représenter de nombreux phénomènes et sont également utilisés pour le rendu d'effets complexes (illumination globale, rendu volumétrique).



Analyse : L'utilisation de voxels rend l'édition manuelle de scènes géométriques 3D très intuitive et permet à des utilisateurs débutants de **sculpter facilement** leurs modèles. Si cette représentation permet de représenter des objets volumiques facilement, elle est beaucoup plus coûteuse en mémoire et nécessite des algorithmes spécialisés pour leur visualisation (que ce soit par extraction de surface ou par lancer de rayons). Les travaux récents de PEYTAVIE *et al.* montrent qu'il est possible de concevoir des outils intelligents ou des algorithmes de simulations qui fonctionnent sur ce type de données et qui permettent ainsi d'augmenter le réalisme d'une scène.

2.2.4 Modèles procéduraux

Jusqu'ici nous avons abordé des représentations qui s'appuient beaucoup sur une structuration des données. Les modèles procéduraux ne sont pas forcément des représentations explicites (*i. e.* elles ne stockent ou ne travaillent pas forcément avec des échantillons du relief) ; à la place, on va s'appuyer sur un **algorithme pour construire** ou reconstruire le terrain à partir de peu d'informations.

Une définition communément admise de ce qu'est un « modèle procédural » est :

> *Un contenu procédural est un contenu numérique (niveau de jeu, modèle 3D, dessin 2D, animation, son, musique, histoire, dialogue, etc) qui a été créé à l'aide d'un algorithme plutôt que par un artiste.*

Cette définition très binaire reflète très mal la réalité. En effet, comment considérer un algorithme qui produit une forêt à partir d'un seul modèle d'arbre ? Et si cet algorithme est paramétrable par un polygone de contrôle qui définit la zone à peupler de végétation ? Et si l'utilisateur peut contrôler la densité ou le type d'arbre ? Les démos graphiques qui sont pourtant intégralement « algorithmiques » sont construites minutieusement dans un but artistique par les *demomakers*.

HALLIWELL part d'un constat simple : **tout contenu numérique est forcément « transformé algorithmiquement »** (que ce soit la transformation projective de l'objet sur l'écran d'affichage, le positionnement ou la déformation qui sont appliqués à l'objet dans une scène, l'éclairage de la surface calculé par le moteur de rendu, *etc*). Parmi ces transformations, certaines peuvent enrichir l'apparence de l'objet en ajoutant des détails géométriques ou sous forme de textures ; d'autres peuvent même construire la géométrie à partir d'une description[9]. Par conséquent, il propose de repenser la définition [154] de ce qu'est un modèle procédural :

> *Rather than thinking of procedural as being a binary thing, why not have it be a continuous scale ? Why not consider content as a collaboration between artists and algorithms, with a definition which allows you to measure "how procedural" content is, in other words, how much of the contribution is algorithmic ?* **Luke HALLIWELL**

9. Ce sont généralement ces objets construits ou enrichis que l'on appelle traditionnellement modèles procéduraux.

Cette définition permet de mettre en avant les qualités des modèles procéduraux « les plus algorithmiques ». L'utilisation de procédures pour générer du contenu permet avant tout de rendre une partie du travail de création **automatique** [10]. Cette automatisation permet à l'infographiste de construire rapidement un contenu artistique (à partir de procédures mathématiques, d'esquisses, ou encore d'une description textuelle). En général, la procédure que le programmeur conçoit est suffisamment générique pour permettre la génération d'**une multitude de variations** d'un même objet. Dans sa forme la plus poussée, l'automatisation est totale et l'algorithme n'attend qu'un nombre aléatoire (qui peut correspondre à un identifiant) pour influencer le processus de création.

Comme l'empreinte mémoire d'un contenu procédural correspond au coût-mémoire de l'algorithme et de ses paramètres, on peut considérer les modèles procéduraux comme **des représentations extrêmement compactes**. Passer d'un modèle procédural à une représentation explicite est relativement simple : il suffit de construire l'objet et de l'échantillonner. À l'inverse, si on a un objet 3D, il est très difficile de retrouver les paramètres d'une procédure qui permette de construire cet objet ; ce type de problème est appelé modélisation procédurale inverse (*inverse procedural modeling*).

De la même manière, une procédure est le seul moyen de décrire des objets avec **une taille et une résolution quasi-infinie** [11]. Il est donc envisageable de représenter des planètes entières avec une précision millimétrique. Bien entendu, la génération de ces modèles procéduraux doit être liée à la visualisation du monde pour ne construire qu'une parcelle de l'univers (*i. e.* l'environnement autour de la caméra). Quand cette génération est faite dynamiquement (au *runtime*) et qu'elle est déterministe (afin de pouvoir « générer/oublier » les modèles 3D dès que cela est nécessaire), on parle de génération « en ligne » (*online*). Un des exemples les plus connus de jeu utilisant ces caractéristiques est la série ELITE datant de 1984 et qui permettait d'explorer des galaxies comprenant des milliers de systèmes solaires alors que ces jeux tenaient en mémoire sur une disquette.

Bien sur, les modèles totalement procéduraux ont des **inconvénients**. Par exemple, lors de la QUAKECON 2013 un spectateur demande son avis à John CARMACK [65] (il est, à ce moment là, directeur technique de iD SOFTWARE et créateur des moteurs ID TECH depuis de nombreuses années) :

> *— One of the things I'm curious about is why you [at iD Software] don't use procedural graphics and procedural geometry more than you do ?*
> *— Ok ! So, procedural graphics has been the way of the future for the last 20 years and I think that I actually have a fairly strong and sound argument, philosophical stance against this. In the end, procedural data is quirky, hard-to-deal-with data compression. And one of the things that we're continuing to get more and more of, is... space. [...]*
> *So it's a good tool for making programmer art but [it is not good] when you want to put it into the hands of the people that are going to make [the assets]... If you're modelling the real*

10. On parle aussi parfois de convertir du « temps de travail artistique » en du « temps de développement informatique ».

11. En pratique, ces algorithmes ont des limites matérielles en fonction de la taille de représentation des entiers ou des flottants.

> *world, you'll laser-scan everything ; you go ahead and say « I'm going to scan this room and I'm gonna get a tera-byte of data, and I'll just render that as this enormous point cloud »... and that's credible even ! It's not we can ship a game like that yet but that's still within sight of something that we can do. And if you want to give it to an artist to create something then they are largely going compositing together different things.*
> *Procedural sources, you use them for your clouds, and your smoke and particle things like that. This was Pixar's camp for a long time : they were creating with analytic procedures rather than textures and* ***that way lost*** [12]*. It was really pretty conclusive that nobody wants to do that. They want to throw 20 layers of effective painting on top of things. And you can still come up with use cases for it but it adds a lot of complexity for a win that for outside of poster-child cases... well... is not there.*
> **John CARMACK**

Si ce constat est relativement pessimiste [13], la réalité lui donne raison sur certains points. Que ce soit en informatique graphique ou dans d'autres domaines comme le son, la génération procédurale présente des inconvénients [115, 332]. Quand elle est utilisée, elle produit souvent des résultats nettement **inférieurs en qualité** à ceux obtenus manuellement. Ainsi les terrains générés procéduralement pour les quêtes optionnelles du jeu MASS EFFECT ont des reliefs extrêmement peu adaptés [14] à l'exploration [8] et les joueurs s'en retrouvent frustrés. De plus, les représentations procédurales sont **difficilement contrôlables** : c'est pourquoi les infographistes retouchent généralement les objets générés dans des logiciels de modélisation traditionnels.

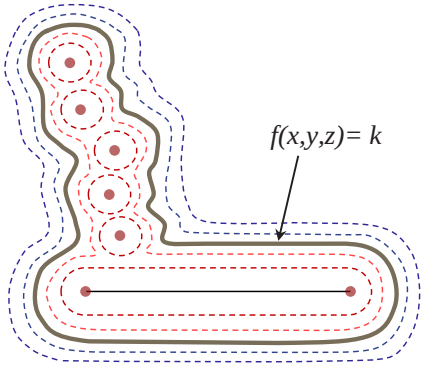
Dans cette section, nous nous intéressons à des structures de représentations procédurales génériques (*i. e.* qui ne stockent pas ou peu de données de la surface du relief). On aborde aussi quelques méthodes d'interpolation qui permettent de reconstruire le terrain entre des échantillons explicites. Les algorithmes de génération de paysage seront détaillés dans la section suivante (Section 2.3). Ces derniers font intervenir des mécanismes de construction qui peuvent être de nature très différentes (des fonctions mathématiques, des agents, des méthodes d'approximation ou d'interpolation ou encore des procédures *ad hoc*).

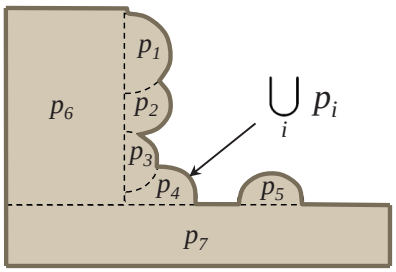
2.2.4.1 Modèles vectoriels

La description vectorielle d'un terrain consiste à stocker quelques informations géométriques du terrain (mais qui ne sont pas forcément des échantillons explicites du relief). Ces informations ne sont pas suffisantes pour reconstruire un terrain par simple interpolation [15] (Fig. 16). Pour visualiser les terrains, il est généralement nécessaire d'utiliser des algorithmes plus complexes d'extraction de surfaces ou de diffusion de contraintes.

12. Il est vrai que de nos jours, PIXAR utilise de moins en moins de textures procédurales, mais de nombreux effets comme la végétation ou les nuages sont encore générés procéduralement. Preuve en est, Iñigo QUÍLEZ (aussi connu sous le pseudonyme de IQ) apporte à PIXAR ses compétences de *demomaker* talenteux depuis plusieurs années.
13. mais aussi paradoxal et ironique, vu que John CARMACK – qui travaille maintenant comme directeur technique chez OCULUS – est considéré aujourd'hui comme un des militants de la réalité virtuelle alors que ce même domaine a été souvent décrit comme « *the wave of the future for the last 20 years* ».
14. Les joueurs se moquent souvent de cette phase de jeu en parlant de « *Mako Climbing*. Cela consiste à gravir (avec difficultés et selon des angles improbables) les montagnes du jeu à l'aide de son véhicule à roues.
15. C'est pourquoi nous classons ces représentations dans les modèles procéduraux.

Un moyen de représenter les objets de manière volumique et continue repose sur les surfaces implicites. On utilise pour cela un ensemble de primitives génératrices (qui sont définies dans une scène 3D) qui créent chacune un champ de densité. En pratique, les champs créés représentent souvent des champs de distance. L'ensemble des champs générés par ces primitives sont combinés par des opérateurs mathématiques (le minimum, le maximum, la moyenne, *etc*). Une version étendue, appelée Blob-Tree est présentée par Wyvill *et al.* [415].





La géométrie constructive solide (*Constructive Solid Geometry*, abrégée CSG) est une technique de modélisation similaire où les primitives représentent généralement des objets simples (cubes, sphères, tores, cylindres) et où les opérateurs correspondent à des fonctions booléennes ensemblistes (l'union, l'intersection, la différence, le complémentaire, *etc*). On peut considérer que la CSG est un cas particulier des surfaces implicites, qui est majoritairement utilisée pour la modélisation de pièces manufacturées.

Des modèles de surfaces implicites plus complexes peuvent être construits à partir des fonctions à bases radiales (*Radial Basis Function* ou RBF). Plus d'informations sont disponibles le rapport de recherche de Coombe [80]. Ce modèle à base de RBF est par exemple utilisé par Pouderoux *et al.* [305] sur des terrains pour lesquels on dispose de peu de données afin de les reconstruire en utilisant une technique appelée *Partition of Unity*.

L'utilisation de fonctions de Hermite à bases radiales (*Hermite Radial Basis Function* ou HRBF) proposée par Macêdo *et al.* [222] permet de représenter des objets 3D de façon plus intuitive. En spécifiant un ensemble de points par lesquels la surface doit passer et les normales en ces points, il est possible de reconstruire une forme 3D [386]. Vital Brazil *et al.* [397] proposent d'utiliser des HRBF variationnelles qui, combinées à une interface d'esquisse, permettent en quelques secondes de modéliser et de représenter des objets 3D dont des terrains.

Toutes ces représentations ne sont pas visualisables facilement : il est nécessaire soit de rendre les images à l'aide de technique de lancer de rayons, soit d'extraire un maillage qui correspond à l'isosurface recherchée.

Un autre modèle vectoriel proposé par Wang *et al.* [402] cherche à représenter des objets volumiques d'échelles différentes à l'aide d'un partitionnement de l'espace grâce à des fonctions de distances signées (*Signed Distance Functions* ou SDF). Ce modèle stocké sous forme de fichiers XML, permet d'obtenir des représentations volumiques avec des variétés géométriques.

Hnaidi *et al.* [162, 163] s'inspirent des courbes de diffusion (*diffusion curves*) présentées par Orzan *et al.* [274] pour le dessin vectoriel. Cette structure permet, à l'aide d'un nombre relativement limité de courbes dont on définit la trajectoire et le profil, de définir le relief d'un paysage sans surplomb. Pour obtenir un maillage de terrain, il est nécessaire d'appliquer un algorithme

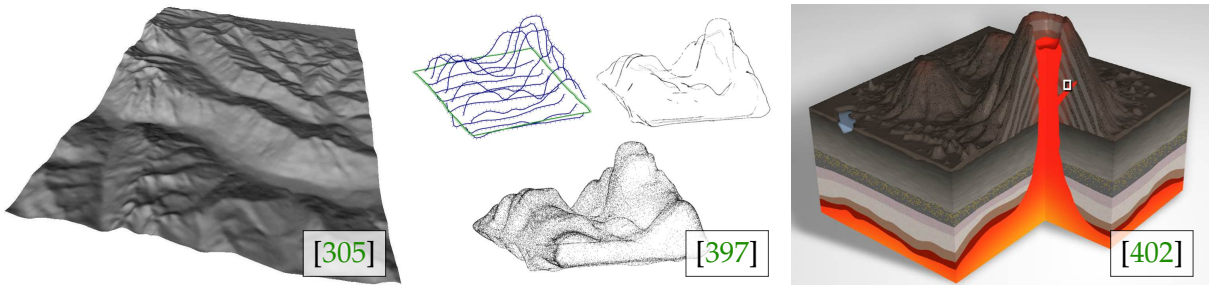
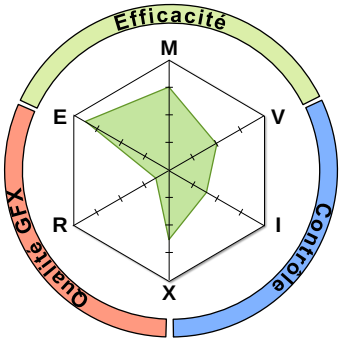


FIGURE 16 : Les modèles à base de surfaces implicites ou dérivées permettent de représenter des objets avec très peu de données. Particulièrement adaptées à représenter des objets lisses, elles sont plus difficilement contrôlables quand il s'agit de construire un relief précis. Ces structures représentant le terrain de manière implicite, il est nécessaire d'extraire un maillage que ce soit pour la visualisation ou pour des calculs de simulations.

de diffusion de la chaleur qui va construire une carte de hauteurs qui respectent les contraintes définies par les courbes. TAKAYAMA *et al.* [367] étendent ce modèle en 3D avec les surfaces de diffusion (*diffusion surfaces*) qui permettent de représenter des objets volumiques 3D. Plus d'informations sont disponibles dans la Section 2.3.4.2 dédiée aux techniques d'esquisses. D'autres modèles vectoriels, plus adaptés à l'édition ou à la décoration de paysages, sont présentés Section 2.3.4.3.



Analyse : Les données vectorielles comme les surfaces implicites ou les courbes de diffusion sont des modèles de terrains **très légers en mémoire**. Avec des modèles vectoriels, il est souvent nécessaire de passer par des algorithmes de visualisation de type extraction de surfaces ou de lancer de rayons. Les surfaces implicites s'adaptent très bien aux objets animés et caractérisés par des surfaces lisses (les liquides, les nuages) et aux changements d'échelles (l'information étant vectorielle). À l'inverse, cette structure nécessite beaucoup de primitives pour contrôler finement la géométrie du relief. En général avec ces modèles, il est impossible d'utiliser des algorithmes de simulation tels quels (il faut d'abord convertir le modèle en un ensemble de voxels ou en un maillage 3D). Quelques travaux plus modernes sur des modèles dérivés comme les HRBF ou les courbes de diffusion permettent à l'utilisateur un contrôle plus intuitif à l'aide de techniques d'esquisses.

2.2.4.2 Interpolations et approximations

Quand on a un nombre de données limité (comme dans le cas des représentations explicites, où on stocke un ensemble d'échantillons du terrain), il est nécessaire d'avoir un processus mathématique qui décrit ce qu'il se passe entre ces données discrètes. On cherche donc à construire une fonction continue C^0 à partir de ces données. Quand la fonction reconstruite passe exactement par tous les points donnés en entrée, on parle d'interpolation. Quand elle essaye de passer au plus proche de tous les points, on parle de *fitting* de surfaces. Quand elle utilise les données

discrètes comme points de contrôle, on parle généralement d'approximation. Si on cherche une fonction qui décrit ce qui se passe en dehors de l'intervalle des échantillons (et non pas entre ces échantillons), on parle d'extrapolation.

En géométrie, il existe de nombreux algorithmes qui cherchent à résoudre ce problème que nous ne détaillerons pas. Parmi les techniques de modélisation qui fonctionnent avec un nombre de points réduit, il est par exemple possible de modéliser des surfaces à l'aide de carreaux de Bézier, de B-Splines ou NURBS.

Interpolations

L'interpolation linéaire en une dimension consiste à construire un ensemble de segments reliant les données d'entrée. On reconstruit donc une fonction C^0 telle qu'elle passe par l'ensemble des données d'entrée ($f(x_i) = y_i$) (Fig. 17).

Ce type d'interpolation s'étend en deux dimensions sur des grilles régulières (Fig. 18) : on parle alors d'interpolation bilinéaire [16]. Ces interpolations sont utilisées aussi bien pour la géométrie que pour les textures. Leur coût en performance est très faible ; des implémentations câblées en dur dans les cartes graphiques sont d'ailleurs utilisées pour les textures.

Des interpolations quadratiques et cubiques existent, et permettent de reconstruire des carreaux lisses entre les échantillons. Ces fonctions, C^1 ou C^2, sont plus coûteuses et nécessitent de connaitre le voisinage les valeurs au voisinage du carreau. Pour reconstruire avec une interpolation bicubique le carreau entre 4 valeurs d'une grille régulière, il est nécessaire de connaître 16 valeurs aux alentours.

Pour obtenir des carreaux lisses mais sans avoir besoin des valeurs aux voisinages, il est possible de fixer certaines contraintes comme les dérivées premières et secondes. À l'aide de polynômes d'interpolations de plus haut degré, les courbes reconstruites seront plus lisses. D'autres techniques qui ne s'appuient pas sur des polynômes existent comme l'interpolation cosinus.

16. Il est intéressant de rappeler que l'interpolation bilinéaire n'est pas une opération linéaire.

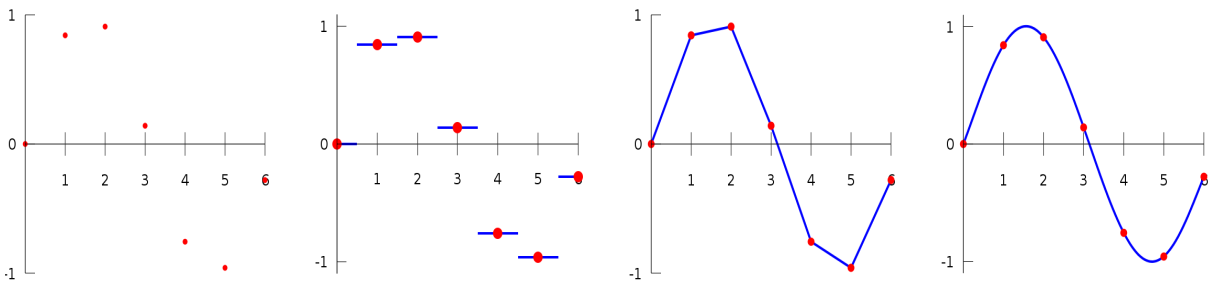


FIGURE 17 : Interpolations d'une fonction à une dimension. De gauche à droite : jeu de données, voisin le plus proche, interpolation linéaire, interpolation cubique.

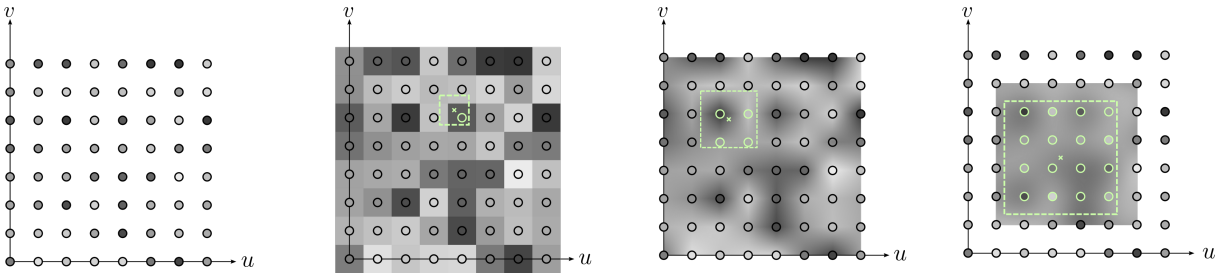
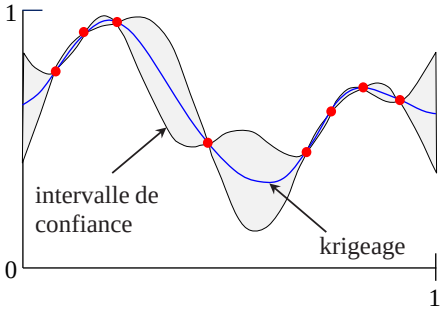


FIGURE 18 : Interpolations d'une fonction à deux dimensions. De gauche à droite : jeu de données, voisin le plus proche, interpolation bilinéaire, interpolation bicubique. En vert pâle : les valeurs utilisées de la fonction pour l'interpolation.

Krigeage

Le krigeage (*kriging*) est une méthode d'interpolation [48] où les valeurs interpolées sont calculées en fonction d'une estimation d'un intervalle de confiance. Pour un ensemble de données d'entrées, on calcule en tout point un intervalle de confiance construit à l'aide d'une loi normale. La fonction reconstruite passe par la médiane de cet intervalle. Contrairement aux méthodes d'interpolation classiques, le krigeage prend en compte non seulement la distance du point recherchée par rapport aux données mais également les distances entre les données deux à deux. Plus ces données sont éloignées, plus l'intervalle d'erreur possible augmente.

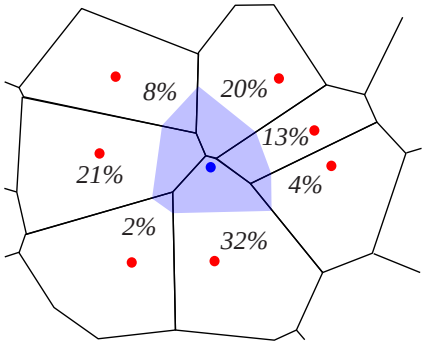
Plusieurs avantages ressortent de l'utilisation du krigeage. Tout d'abord, il s'agit d'une méthode d'interpolation et non d'approximation (Fig. 19 gauche). Ensuite, il n'y a aucune topologie à gérer (on travaille juste sur un ensemble de points dans un espace de N dimensions). Enfin, si on souhaite contrôler le processus, on peut souhaiter rajouter un point de contrôle (donc un point de donnée d'entrée) : si ce dernier se trouve sur la courbe reconstruite, il ne fera que diminuer l'estimation de l'erreur mais ne changera pas la fonction reconstruite.



Malheureusement, le calcul du krigeage est plus complexe qu'une simple interpolation linéaire (cela revient à calculer des inverses de matrices n par n avec n correspondant au nombre d'échantillons). Pour contrôler l'édition de grands terrains, il est possible de décomposer un terrain en plusieurs patchs, chacun géré par le krigeage [46].

Pondération inverse à la distance et voisinage naturel

La pondération inverse à la distance (*inverse distance weighting*) est une méthode d'interpolation proposée par SHEPARD en 1968 [342] (Fig. 19 centre). Pour trouver l'élévation d'un point quelconque, on va calculer la contribution de chaque point connu et pondérer cette contribution en fonction de la distance. Généralement ce coefficient de pondération est calculé comme l'inverse de la distance entre les points, le tout élevé à une certaine puissance.



Le voisinage naturel (*natural neighbor interpolation*) est une autre méthode d'interpolation proposée par SIBSON en 1981 [345]. La méthode consiste à étudier le diagramme de Voronoï construit par les échantillons. Pour calculer l'élévation d'un point interpolé que l'on rajoute sur la surface, on simule son insertion dans le diagramme de Voronoï. Ajouter ce point va créer un nouvelle cellule virtuelle qui va repousser les cellules voisines (et donc s'approprier une partie de leur aire). L'élévation du point interpolé est calculée par la somme des contributions des points voisins. Chaque contribution correspond à la valeur d'altitude du point voisin pondérée par le ratio entre l'aire subtilisée à la cellule voisine, et l'aire de la nouvelle cellule insérée. Par conséquent, cet algorithme permet de différencier des contributions de points qui se trouvent à la même distance.

Le voisinage naturel est très utilisé pour reconstruire des terrains où les échantillons sont distribués de manière non-homogène (Fig. 19 droite). Beaucoup de travaux se sont intéressés à améliorer cette technique. Par exemple, LEDOUX et GOLD [209] généralisent l'algorithme pour travailler avec des données dans un espace à N dimensions (*e. g.* des données 3D+temps) et proposent un algorithme plus efficace que le voisinage naturel original en terme de complexité de calculs.

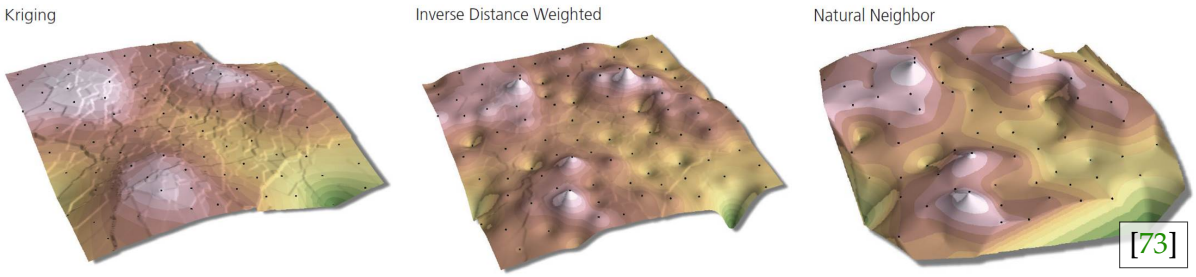


FIGURE 19 : Des méthodes d'interpolations plus complexes (respectivement de gauche à droite : krigeage, distance inverse, voisinage naturel) sont très utilisées en géographie pour reconstruire des données réelles incomplètes. Ces méthodes sont souvent coûteuses en temps de calculs et peu adaptées pour reconstruire un terrain lisse en temps-réel.

2.2.4.3 Modélisation fractale

Avant l'étude des fractales, la nature n'était vue qu'au travers de la géométrie euclidienne. On peut résumer cette vue par une citation du peintre CÉZANNE :

> *Tout dans la nature se modèle sur la sphère, le cône et le cylindre, il faut apprendre à peindre sur ces figures simples, on pourra ensuite faire tout ce qu'on voudra.* **Paul CÉZANNE**

Benoît MANDELBROT, mathématicien français du XXième siècle, invente le concept de fractales (en s'appuyant sur les théories de Hausdorff et de Julia) : ce sont des objets dont la complexité

visuelle est produite par un processus récursif[17]. Une fractale est nécessairement définie par une règle impliquant la répétition d'un motif au travers d'une homothétie (un changement d'échelle) : on parle également d'auto-similarité (l'ensemble du motif est semblable à une de ses parties) ou de symétrie par dilatation. MANDELBROT remarque également [225] que le relief des terrains ressemble aux fonctions fractales qu'il manipule, il parle d'ailleurs de l'ensemble du domaine des fractales comme de **la géométrie de la Nature** expliquant que :

> *Les nuages ne sont pas des sphères, les montagnes ne sont pas des cônes, les côtes maritimes ne sont pas des cercles, et les éclairs ne sont pas des lignes droites.*
>
> **Benoît MANDELBROT**

Benoît MANDELBROT étend en deux dimensions le concept de fonctions fractales et obtient des surfaces qui ressemblent à des reliefs montagneux (y compris quand ces objets sont vus à l'envers). Par la suite, son étudiant Forest Kenton MUSGRAVE – dont le parcours est plus lié à l'infographie naissante qu'aux mathématiques – continuera à développer de nombreux travaux qui sont à la frontière entre fractales et modélisation de terrains. De nombreuses informations sur les fractales ont été compilées par BARNSLEY dans un livre [19] et un cours *SIGGRAPH* [20].

Un des moyens de construire une fonction fractale consiste à utiliser un processus mathématique chaotique appelé fBm ou *fractional Brownian motion* (du nom de son découvreur). Une fBm est une fonction mathématique qui possède plusieurs propriétés : elle est statistiquement invariante selon la translation (ce qui n'est évidemment pas le cas des terrains réels) ; en une dimension, elle est statistiquement symétrique selon un plan horizontal. Dans le cas général, elle est isotrope (le processus n'a pas de direction privilégiée) et stationnaire (le processus ne change pas au court du temps). Enfin, elle est auto-similaire, ce qui est la caractéristique des objets fractals (c'est le seul type de mouvement brownien qui l'est, d'où le terme de *fbm*).

Ce processus mathématique des fBm est guidé par une variable appelée coefficient ou exposant de Hurst qui est liée à la notion de dimension fractale. Cette dernière mesure de manière non-entière comment un objet fractal va remplir l'espace : par exemple, une courbe, qui est pourtant un objet de dimension 1, peut, si elle est récursivement subdivisée selon le schéma de Peano, remplir l'espace (c'est-à-dire que pour un point testé, on peut calculer le nombre de subdivisions nécessaires pour que la courbe fractale passe par ce point). On dit alors que cette courbe remplit le plan (*space-filling curve*) ou que sa dimension fractale est de 2. L'étude du coefficient de Hurst est nécessaire pour analyser la forme de nombreux phénomènes fractals dont la forme des réseaux hydrographiques [191].

On peut donc considérer que la géométrie fractale permet avant tout de modéliser des surfaces rugueuses :

> *Toutes les sensations que nous avons ont à tour de rôle été domptées par la science : le poids par la mécanique, le chaud et le froid par la thermodynamique, le brillant par l'optique, le son par l'acoustique. Il n'y avait pas de mesure numérique de la rugosité, perçue par l'œil et la main, jusqu'à ce que j'en publie une en 1984. J'ai trouvé dans les fractales l'objet fondamental*

17. Intuitivement, une fractale correspond davantage à une procédure algorithmique infinie qu'à un objet à une résolution donnée. C'est pourquoi nous considérons que ces objets sont fortement liés à la notion de « modèles procéduraux ».

de la rugosité comme la sinusoïde est l'objet fondamental de la lumière et du son.
Benoît MANDELBROT

Un concept proche est celui de signal $1/f^{\beta}$ qui représente une fonction dont la densité spectrale de puissance suit une loi exponentielle inverse. Bien qu'un bruit soit un signal aléatoire spatialement, il suit généralement cette caractéristique statistique $1/f^{\beta}$. On parle souvent de la couleur du bruit en lieu et place de la valeur de β. Un bruit blanc ($\beta = 0$) possède une fonction de densité spectrale linéaire : on retrouve toutes les fréquences avec la même amplitude. On parle de bruit rose quand $\beta = 1$. Enfin lorsque $\beta = 2$, on appelle ce signal un bruit brownien ou brun [18]. Les processus fBm génèrent souvent ce type de signal.

Plusieurs processus de génération procédurale de terrains présentés dans les années 70 et 80 construisent des terrains fBm (généralement des surfaces $1/f^2$). Une fBm peut être construite par un algorithme de marche aléatoire (*random walk*), par une somme de sinus (utilisée dans la synthèse spectrale), par une somme de fonctions de bruits (utilisée dans la synthèse de bruit) ou encore par une somme de signaux en dents de scie (utilisée dans les méthodes de subdivision). Une présentation détaillée (et une analyse mathématique) de ces processus créateurs de surfaces fBm est disponible dans la thèse de MUSGRAVE [253].

Si Mandelbrot pensait que les fractales permettaient de modéliser des paysages réels, il semble que de nombreux travaux plus récents contredisent cette hypothèse. En 1990, une analyse de LEWIS [215] montre que les terrains ne sont pas des fractales : il existe bien des similarités au niveau du relief entre certaines échelles, mais pas à toutes les échelles, et pas pour tous les types de paysages. Malgré tout, il pense que d'autres phénomènes naturels (en particulier les nuages) peuvent être modélisés avec cette géométrie fractale. MUSGRAVE [253, 255] montre qu'un terrain fBm, donc statistiquement homogène, n'est pas forcément une caractéristique intéressante. En effet, les phénomènes d'érosion qui forment les rivières et les crêtes sont des éléments anisotropes qui ont une structure hiérarchique ; ils sont donc difficiles à modéliser avec des fractales.

Au final, on a d'ailleurs **peu d'idées fiables sur les processus qui forment les montagnes**. On sait que la tectonique joue un rôle, de même que les érosions hydriques et glaciaires. Une analyse de plusieurs travaux contradictoires est d'ailleurs présentée par MOLNAR en 2003 dans *Nature* [247]. Une des rares règles qui semblent régir la description des paysages correspond à la première loi de la géographie énoncée par TOBLER (*Tobler's first law of geography*) :

Everything is related to everything else, but near things are more related than distant things.
Waldo TOBLER

Cette dernière indique que deux objets proches ont plus de chances d'interagir ensembles que deux objets éloignés. Il en découle que l'ensemble des mécanismes de création ou de modification du relief ont souvent un effet local relativement homogène.

18. jeu de mot anglais puisque *brown* est le début de *Brownian*

2.3 Génération et édition de terrains

Après s'être intéressé aux modèles de terrains qui décrivent comment représenter et stocker des paysages 3D, on cherche à décrire les outils et les procédures pour créer, modifier ou générer automatiquement ces terrains.

2.3.1 Génération ex-nihilo

On retrouve deux cas majeurs d'utilisation de la génération procédurale : soit comme base d'inspiration, soit comme modèle de représentation-pivot pour représenter certains types d'environnements.

Dans le premier cas, on utilise ces algorithmes comme un moyen semi-automatique pour créer du contenu. Si on conçoit des méthodes de génération dans ce but, on cherche avant tout à permettre à l'utilisateur d'exercer un contrôle sur la génération. Ce dernier veut que les paramètres de génération soient le plus intuitifs possibles (qu'ils représentent par exemple le nombre de continents d'une planète) et que leur manipulation et leurs variations soient aussi prédictibles que possible (si l'utilisateur change le paramètre de température, il veut voir de la neige apparaître). Généralement ces algorithmes permettent à l'utilisateur de construire très rapidement un premier résultat qu'il pourra exporter et retoucher dans un autre logiciel.

Dans le deuxième cas, on utilise ces algorithmes comme une partie intégrante des moteurs de rendu. On dit que la génération du monde se fait dynamiquement ou « en ligne » quand le terrain est construit alors que la caméra bouge dans le monde. Ce type de moteur graphique permet de rendre des mondes immenses avec une grande échelle de résolution (les algorithmes utilisés se basant souvent sur des mécanismes de pseudo-niveaux de détails) qui permet de représenter à la fois des planètes et des petits reliefs de l'ordre de quelques centimètres. Dans ce cas d'utilisation, les algorithmes doivent avant tout être performants (aussi bien en temps de calculs qu'en coût-mémoire) et permettre de construire un terrain localement et à plusieurs résolutions. En général, ces algorithmes sont contrôlés par un nombre très restreint de paramètres (généralement uniquement une graine d'aléatoire (*random seed*)).

2.3.1.1 Schémas de subdivision

Parmi les premières techniques de génération procédurale de terrains, plusieurs travaux ont exploré les méthodes de subdivisions (Fig. 20). Ces processus cherchent à subdiviser et raffiner itérativement la fonction de relief d'un terrain pour y ajouter de plus en plus de détails. Tous les algorithmes de subdivision produisent des terrains qui sont fractals mais seules certaines techniques permettent d'obtenir les caractéristiques statistiques d'un terrain fBm.

L'algorithme le plus connu est celui du *midpoint displacement* [121, 122, 223] dans lequel on calcule progressivement des grilles régulières d'élévations de plus en plus fines, dont les nouvelles élévations sont calculées en fonction des points voisins déjà existants, et d'une valeur de

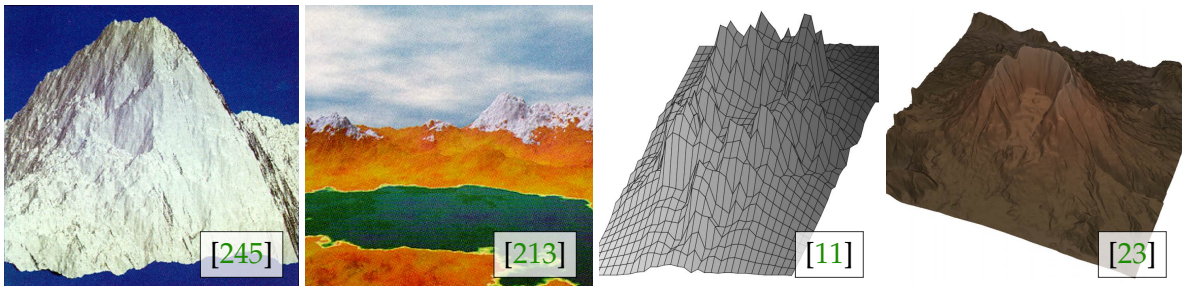


FIGURE 20 : Les méthodes à base de subdivisions sont très rapides et permettent – avec des techniques plus avancées – d'introduire une notion de structure hiérarchique.

perturbation. Cette dernière décroît obligatoirement avec les itérations afin de ne générer plus que des hautes fréquences correspondant aux détails.

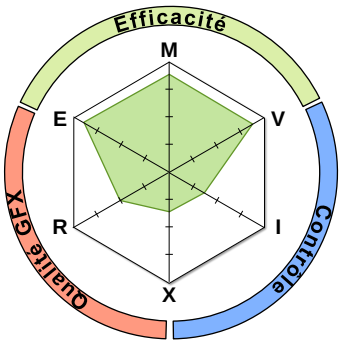
Plusieurs travaux [213, 224, 245] ont proposé de changer la configuration de sélection des voisins (on parle de schéma de subdivision en triangle, en carré, en losange-carré) ou la manière dont est calculée la valeur de perturbation. Le schéma de subdivision est important puisqu'il peut introduire des artefacts visuels directionnels. Il est également possible d'utiliser une grammaire pour encoder les subdivisons du terrains [10, 11]. Dixon *et al.* [99] étudient les algorithmes de subdivision sur plusieurs structures de données (triangles, quads, hexagones) et proposent une structure mixte à base de polygones qui permet d'utiliser plusieurs formes en même temps.

Il est possible d'utiliser les schémas de subdivision pour aider à créer une structure hydrologique. KELLEY *et al.* [185] proposent en 1988 de construire un réseau hydrographique représentant un bassin-versant en modifiant l'algorithme de *midpoint displacement*. En démarrant d'une simple trajectoire où l'on opère des subdivisions et des embranchements, on parvient à créer un réseau drainant. Certaines méthodes s'intéressent à créer des rivières par subdivision de leur trajectoire comme dans l'approche de PRUSINKIEWCZ et HAMMEL [308]. Plus récemment, DERZAPF *et al.* [98] ont proposé de construire une planète entière à l'aide de subdivisions tout en garantissant la construction et l'écoulement de rivières jusqu'à la mer.

Il est également possible de construire un terrain en définissant des surfaces fractales. Plusieurs travaux plus génériques sur la manipulation des fractales existent dans la littérature [143, 152, 381] et certains ont essayé d'appliquer ces méthodes fractales pour ajouter des détails sur un terrain [162]. Il est également possible de modifier certains de ces algorithmes de subdivision pour permettre à un utilisateur de contraindre le résultat : c'est ce que font SZELISKI et TERZOPOULOS [366] qui couplent une représentation à base de splines avec un modèle fractal.

BELHADJ propose en 2007 [22, 23] un système de reconstruction qui s'inspire des algorithmes de subdivision. Au lieu de construire ou reconstruire un terrain avec des données basse résolution (*i. e.* une grille grossière), il démarre avec des points caractéristiques qui serviront de contraintes d'élévation. Ces points de contraintes peuvent être placés à la main (dans le cadre de la génération) ou calculés automatiquement depuis un image réelle (dans le cadre d'une compression/reconstruction).

Analyse : les méthodes à base de subdivision permettent de représenter des mondes immenses en raffinant au fur et à mesure un relief de base. Les coûts-mémoire et de visualisation sont donc très bons et ces méthodes sont parmi les seules à pouvoir générer des corps de taille planétaire. Si ces méthodes sont généralement simples à comprendre, elles sont plutôt difficiles à contrôler par un utilisateur. Plusieurs extensions ont été proposées pour utiliser les différents niveaux de subdivision pour **structurer hiérarchiquement les reliefs** avec, par exemple, des réseaux hydrographiques.



2.3.1.2 Synthèse spectrale (synthèse par Fourier inverse)

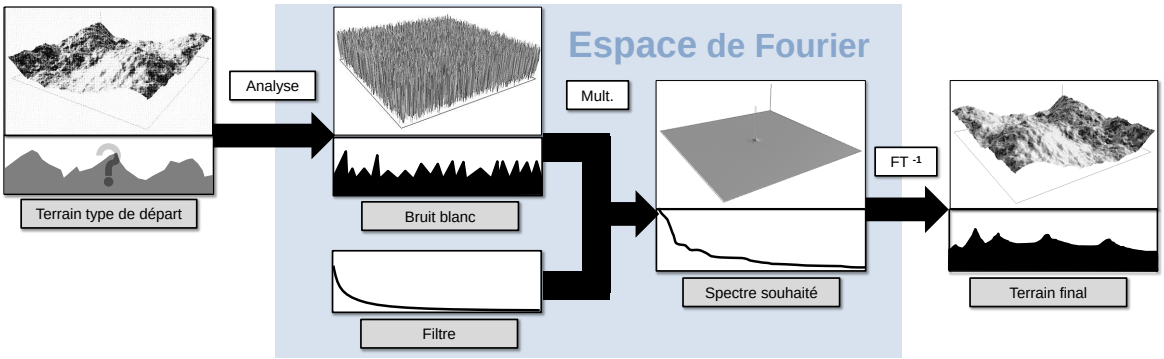


FIGURE 21 : Générer un terrain à partir de son spectre est possible en utilisant une transformée de Fourier inverse. Le spectre artificiel que l'on souhaite créer est construite en multipliant un filtre passe-bas (de la forme souhaitée, souvent construit après des analyses du type de relief voulu) avec un spectre de bruit blanc (où toutes les fréquences sont présentes). Le terrain créé possède les caractéristiques spectrales souhaitées (généralement un bruit brun) et parce qu'il est construit par Fourier inverse, ce morceau est pavable (*tileable*).

Le principe de l'algorithme original [339] est d'obtenir des terrains qui soient fBm c'est-à-dire dont la densité du spectre est proportionnelle à l'inverse de la fréquence à une puissance donnée soit $1/f^{\beta}$. Quand β vaut 0, on obtient un bruit blanc ; quand il vaut 2, on obtient un bruit brun ou brownien.

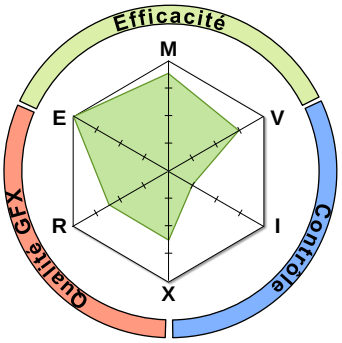
Si l'on visualise le spectre de puissance avec une échelle logarithmique, $-\beta$ correspond à la pente de la fonction. L'objectif est alors de créer un bruit dont le spectre obéit à une distribution voulue dans l'espace de Fourier puis par transformation inverse, d'obtenir le signal qui correspond à ce spectre.

Pour cela, on commence par créer un bruit blanc (un spectre de puissance plat où toutes les fréquences sont représentées avec la même amplitude). La transformée de Fourier d'un bruit blanc donnant une fonction constante, on peut tout de suite se placer dans l'espace de Fourier. On va déformer ce signal dans l'espace des fréquences à l'aide d'un filtre passe-bas pour obtenir une nouvelle courbe de puissance (Fig. 21). Ce filtre va diminuer l'amplitude des hautes

fréquences qui correspondent aux détails du terrain, alors que les basses fréquences de grandes amplitudes correspondant aux reliefs à grande échelle seront laissées intactes. Une fois le signal filtré, il est possible de revenir dans l'espace image en faisant la transformée de Fourier inverse. Le signal obtenu ressemble alors à un terrain [52].

Le même type de méthode peut être utilisé avec d'autres filtres pour simuler d'autres objets naturels (Fig. 22). Le filtre de Pierson-Moskowitz permet de modéliser des vagues [216, 231]. SAKAS propose, lui, de modéliser des gaz en mouvement [334]. Il est également possible d'utiliser la synthèse de Fourier inverse en espace image : par exemple, LANGER *et al.* [205, 206] cherchent à ajouter une animation de neige tombant du ciel, à une scène virtuelle.

Plus récemment, des techniques de synthèses de bruits s'appuyant sur des décompositions et des manipulations en espace de Fourier ont été mises au point. Elles seront abordées dans la prochaine section.



Analyse : les techniques de synthèse spectrales sont très intéressantes car elles sont suffisamment **génériques pour résoudre de nombreux problèmes** (génération du relief, de la forme des nuages ou des vagues, effets météorologiques animés, ...). De plus, ces techniques ne sont pas très coûteuses en terme de calcul (surtout avec de nombreuses bibliothèques de code performantes disponibles) même si elles nécessitent quelques bases mathématiques pour les comprendre et les utiliser. Par contre, ces méthodes sont difficilement combinables avec d'autres techniques bien que les travaux récents sur les bruits (comme le *Local random phase noise* [146]) ouvrent de nouvelles possibilités en combinant à la fois contrôle spatial et fréquentiel.

2.3.1.3 Synthèse par bruits

Il y a de nombreuses méthodes stochastiques [106] pour générer des terrains qui se basent sur des approches fonctionnelles. Ces méthodes cherchent à construire une fonction mathématique (planaire ou non) dont l'évaluation est locale et qui imite le relief d'un terrain. Pour une introduction sur les méthodes de bruits, cf. le cours de PATEL [282].

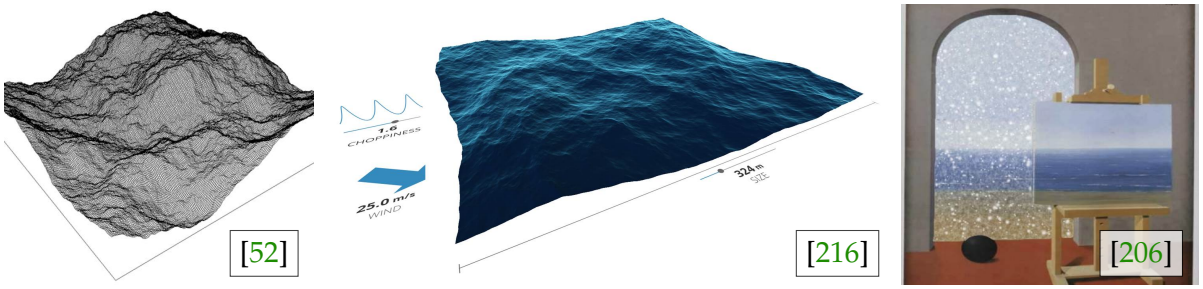


FIGURE 22 : La synthèse spectrale est une technique suffisamment générique pour modéliser plusieurs types de phénomènes (terrains, océans, nuages).

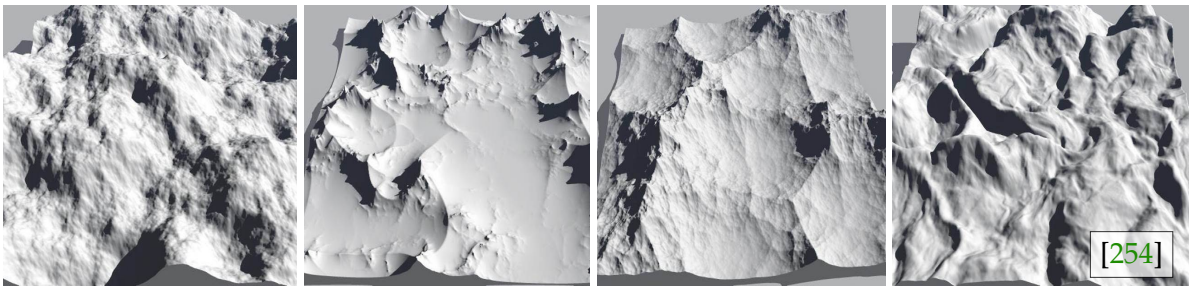


FIGURE 23 : MUSGRAVE propose plusieurs transformation pour modifier les sommes de bruits de Perlin (gauche) que ce soit en modifiant la dimension fractale (centre gauche), en combinant son utilisation avec un bruit cellulaire (centre droit) ou en déformant le domaine (droite).

En particulier, le bruit de Perlin introduit en 1985 [288, 289] permet non seulement de générer des cartes de hauteurs cohérentes mais également des textures. En sommant des harmoniques de bruits d'amplitude et de fréquences différentes, on parvient à générer des cartes de hauteurs qui miment la réalité. Ce bruit a été amélioré plusieurs fois par PERLIN. En 2002, il présente une version utilisant des polynômes qui lissent le bruit lors de l'interpolation [291]. Il propose en parallèle un autre bruit, le Simplex noise qui est plus robuste et dont la complexité est bien meilleure quand on le calcule dans un espace de dimension élevée [153, 290].

Il existe de nombreux travaux dérivés [187] autour du bruit de Perlin : à l'aide de quelques transformations que l'on applique sur la fonction d'élévation, on peut parvenir à générer des paysages plus complexes. L'utilisation de bruits de plus hautes dimensions permet également d'animer ces mêmes structures (par exemple, pour représenter des nuages en mouvement). PERLIN propose avec NEYRET un bruit combiné avec une fonction d'advection [292] qu'il est possible de calculer en temps-réel [13]. PARBERRY [275] propose d'utiliser la programmation dynamique pour diminuer les temps de calcul sur CPU.

Plusieurs types de bruits existent : bruit par valeurs [219], bruit par gradient [288], bruit de Simplex [153], bruit cellulaire [413], bruit par ondelette [79], bruit de Gabor [198], bruit de Quilez [314], Voronoise [317]. En 2010, un état de l'art spécifiquement dédié aux aspects théoriques de construction des bruits est proposé par LAGAE *et al.* [199]. Un rapport classifiant ces différents types de bruits est disponible ici [174]. Si certains de ces bruits sont plutôt utilisés pour s'appliquer sur une surface, d'autres sont spécialement étudiés pour générer des textures volumiques ou *solides* [214, 286]. Quelques bruits sont construits en manipulant l'espace des fréquences (*Spot noise*, bruit de Gabor, *Space Convolution Noise*). Récemment, GILET *et al.* [146] ont proposé un bruit (*local random phase noise*) contrôlable à la fois dans l'espace spectral et dans le domaine spatial. Enfin, plusieurs de ces bruits [63, 97, 391] permettent de facilement construire des champs de vecteurs (*LIC noise*, *Spot noise*).

Tout comme dans les méthodes de subdivisions, le contrôle de la synthèse de bruits n'est pas chose aisée : le sens des paramètres est parfois contre-intuitif, et chaque modification des paramètres provoque des changements globaux sur l'ensemble de la fonction. De plus, comprendre ce que l'artiste souhaite, et trouver une fonction qui corresponde à ce qu'il attend est une tâche complexe. À contrario, l'utilisation de ces algorithmes permet de générer de véritables terrains

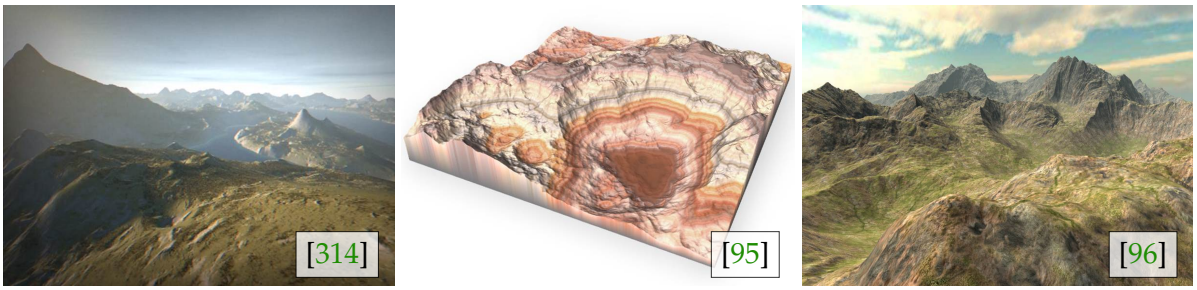


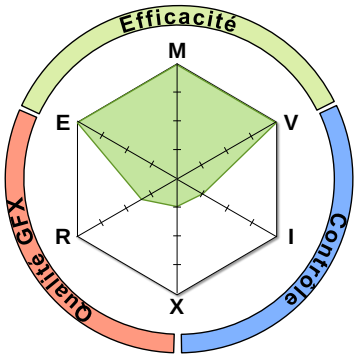
FIGURE 24 : Les bruits peuvent être façonnés pour donner des paysages très réalistes qui peuvent même imiter certaines formes d'érosion.

infinis, et de façon dynamique (on ne génère le terrain que près de la caméra). Comme chaque point est calculé indépendamment de ses voisins, ce type d'opération peut être facilement déporté sur le GPU [156, 333, 346].

Il existe des travaux spécifiques concernant les bruits appliqués à la création de paysages. En particulier, MUSGRAVE *et al.* [106, 254, 255] proposent de modifier les propriétés fractales des fonctions de bruits afin de modéliser des terrains composés de différentes zones comme des plaines et des montagnes. Ils construisent des fonctions multi-fractales c'est-à-dire dont la dimension fractale varie dans l'espace (Fig. 23). SAUPE propose d'étendre ce type d'opération de variation de la dimension fractale à l'aide de sommes de bruits de Perlin [340]. La configuration précise et manuelle d'un bruit permet d'imiter certains reliefs de manière réaliste comme ZAAIJER le fait avec les dunes de sables [419].

VAN LAWICK et JENSE [390] essayent de contrôler des systèmes multi-fractals en analysant des terrains réels : il obtient plusieurs valeurs caractéristiques qu'il utilise comme paramètres pour générer des terrains bruités. Plus récemment, PARBERRY [276] montre comment modifier le bruit par valeurs pour obtenir un histogramme statistiquement proche de données réelles. Il fait de même [277, 278] avec un bruit de gradient et avec les données de distribution de gradients d'une région de l'Utah.

Afin d'éditer et de contrôler les fonctions de synthèse de bruits, plusieurs solutions ont été mises en place. SCHNEIDER *et al.* [341] se concentrent sur le rendu temps-réel et le paramétrage de ces fonctions. DE CARPENTIER et BIDARRA [95, 96] proposent d'utiliser des pinceaux (des outils de dessins 2D) qui configurent les bruits en temps-réel grâce à une implémentation GPU. De plus, avec des fonctions de déformation du domaine, ils montrent qu'il est possible d'imiter le relief structuré autour de vallées, avec un bruit (Fig. 24).



Analyse : Les représentations de terrains fonctionnelles par bruits sont **les méthodes reines de la génération procédurale**. Elles ont un coût-mémoire presque idéal (quelques paramètres de générations globaux) et permettent de générer localement le terrain (contrairement aux méthodes de subdivision). Leur résolution peut être infinie et elles sont aussi bien utilisées pour modéliser des chaînes montagneuses que des détails au sol. Mais leur contrôle est extrêmement difficile puisque les changements sont généralement globaux (un changement de paramètre modifie l'intégralité de la fonction). De plus, elles ne sont pas très expressives (les fonctions de bruit 2D sont limitées à générer des terrains planaires, elles sont difficiles à coupler avec d'autres méthodes). Enfin, ces représentations étant fonctionnelles, elles ne permettent pas d'avoir les informations de voisinages d'un point, ce qui complique beaucoup la modélisation de l'hydrologie sur un terrain.

2.3.1.4 Intelligence artificielle

On regroupe dans ces méthodes des solutions de génération procédurale qui peuvent s'appliquer de manière générique à plusieurs familles de problèmes (modèles 3D, sons, histoires, règles de jeux). On parle également de *search-based procedural content modeling*. Un état de l'art spécifiquement ciblé sur ces méthodes a été écrit par Raffe *et al.* [319]. Plusieurs de ces techniques sont également utilisées pour générer des terrains (Fig. 25).

Une méthode de création d'îles à l'aide d'agents est proposée par Doran et Perberry [100]. Si le contrôle n'est pas direct, l'algorithme est relativement intuitif : on laisse agir plusieurs agents qui modifient localement le terrain autour d'eux. Leurs règles de déplacements et leurs actions sont dépendantes de leur rôle : dessiner le contour de l'île, adoucir le relief, créer des plages, placer des montagnes ou tracer des rivières. Cabezas et Thompson [62] cherchent à sculpter interactivement un terrain en guidant un essaim d'agents qui transforment le relief.

Des approches évolutionnaires sont proposées pour générer des terrains. Ong *et al.* [271, 338] proposent d'utiliser de décomposer le problème de la génération de terrains en deux sous-problèmes : tracer des silhouettes de patchs de terrains et calculer les élévations de chacun de ces patchs. Ces deux problèmes sont encodés sous formes de chromosomes qui sont mutés et combinés avec un algorithme génétique pour obtenir de nouveaux terrains similaires. De la même manière, Ashlock *et al.* [10, 11] essaient de faire varier des terrains encodés sous la forme d'une grammaire de subdivisons (à base de L-systèmes) en utilisant un algorithme génétique. Raffe *et al.* [318] utilisent aussi des algorithmes évolutionnaires. Ils découpent le terrain en une grille de patchs reliés par une fonction d'interpolation. Chaque bloc de terrain est sélectionné aléatoirement dans une base de données ; combiner deux terrains revient à sélectionner les blocs de l'un ou de l'autre.

Frade *et al.* [123, 124, 125, 126, 127, 128, 129, 130] proposent d'utiliser la programmation génétique (l'évolution d'arbres de fonctions) pour générer procéduralement le relief de terrains. On définit donc le relief comme une fonction mathématique manipulant des briques qui peuvent

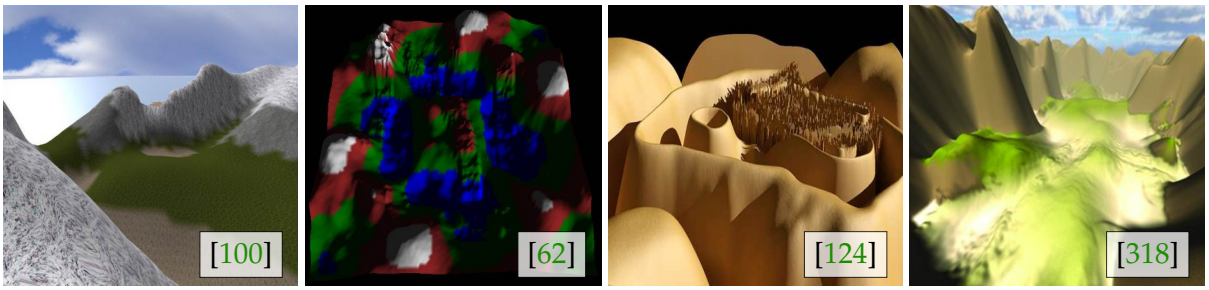
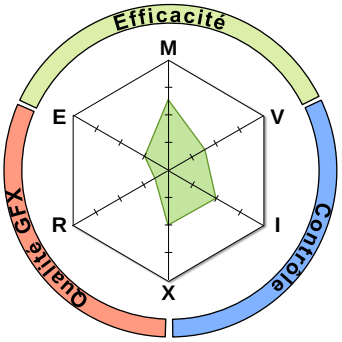


FIGURE 25 : Les méthodes à base d'intelligence artificielle (algorithmes évolutionnaires, agents, optimisations, ...) sont relativement récentes. Les terrains produits ont très souvent un aspect artificiel et de nombreuses techniques de bruits surpassent ces résultats.

correspondre à des fonctions d'addition, de multiplication, des sinus et cosinus ou encore des logarithmes.

WALSH et GADE [401] utilisent une interface graphique web pour laisser l'utilisateur guider un algorithme génétique vers le résultat souhaité.



Analyse : de nombreux travaux récents et effectués au sein de la communauté travaillant sur l'intelligence artificielle (et les techniques d'optimisations stochastiques) concernent la génération de reliefs. Ces articles sont souvent publiés dans des journaux et présentés à des conférences qui ne relèvent pas de l'informatique graphique. Les résultats sont souvent d'une qualité inférieure aux autres familles de méthodes de génération mais certaines conclusions tirées de leurs approches peuvent permettre d'améliorer des **algorithmes de génération de contenu procéduraux plus généralistes**. Il est pourtant intéressant de noter que ces travaux cherchent souvent à mêler à la fois une production artistique et visuelle et des contraintes de *gameplay* : c'est pourquoi plusieurs de ces travaux ont récemment influencé le développement de certains jeux-vidéos.

2.3.1.5 Faulting

Le *poisson faulting* est une des premières techniques de génération procédurale de paysage qui ait été créée. Elle fonctionne de manière itérative : sur une grille régulière, on fracture le terrain le long d'une droite tirée aléatoirement. Chaque demi-plan est surélevé (ou abaissé) d'une valeur de décalage. Cette valeur décroît progressivement afin de ne rajouter plus que des détails (Fig. 26).

Il existe plusieurs variations sur la façon d'élever les demi-plans [379]. Chacune de ces variations donne des résultats légèrement différents. Ce même processus peut s'étendre à la génération d'une planète [53, 109]. Il suffit de partir d'une sphère et de sélectionner des demi-plans à décaler. Il a été montré [223, 253, 400] que ce processus crée un terrain qui respecte les propriétés statistiques d'un fBM.

Kamal et Sarwar Uddin [181] présentent une variation du *faulting* pour créer des montagnes individuelles dont on paramètre la position du pic et la hauteur finale. Là encore, l'algorithme fonctionne itérativement : à chaque étape, à l'aide d'un ensemble de demi-droites tirées aléatoirement, on construit une segmentation de notre paysage (différente à chaque étape). Un ensemble de ces régions est sélectionné (incluant toujours celle du pic) pour être légèrement élevé.

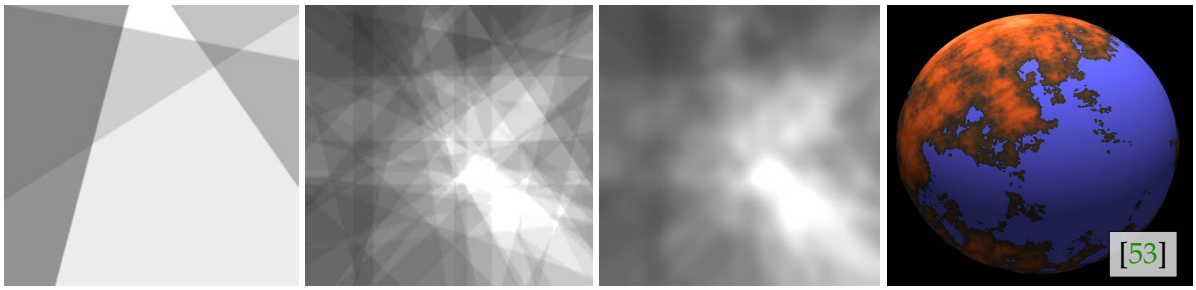
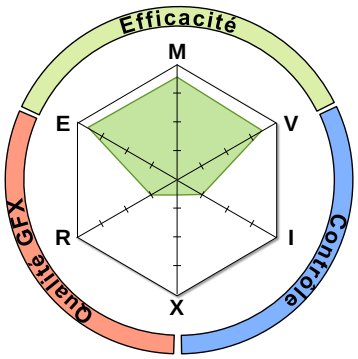


Figure 26 : Les techniques de *faulting* sont parmi les plus vieilles techniques de génération de terrains. Leur facilité d'implémentation couplée à la possibilité de les utiliser pour générer des planétoïdes continue de rendre ces méthodes populaires auprès des développeurs amateurs

Plus récemment, des travaux se sont intéressés à ré-imaginer le *faulting* d'un point de vue simulationniste. Au lieu d'imaginer un processus mathématique de fractures qui ne repose sur aucune réalité, ces travaux cherchent à reproduire le mouvement des plaques tectoniques. Ces techniques sont présentées dans la Section 2.3.3.1.

Analyse : Les techniques de *faulting* sont **les premières méthodes** de génération procédurales de terrains ; leur conception remonte à la fin des années 70. Peu intuitives, les techniques originelles sont également dépassées en terme de performances quand il s'agit de générer un terrain fBm (par rapport aux méthodes de subdivision et aux représentations fonctionnelles par bruits). Aujourd'hui, de nouvelles techniques se rapprochant de la simulation tectonique sont conçues et tendent vers un plus grand réalisme des scènes construites.



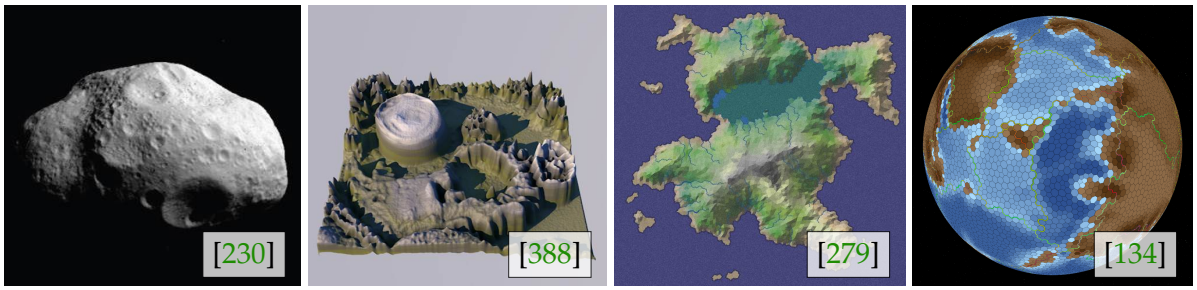


FIGURE 27 : D'autres méthodes (présentées dans des articles, des livres d'informatique graphique, par des amateurs ou des scientifiques d'autres domaines) existent pour modéliser des reliefs.

2.3.1.6 Méthodes *ad hoc*

Il existe de nombreuses méthodes de génération de terrains qui n'entrent pas forcément dans une catégorie ou une famille d'algorithmes (Fig. 27). De nombreux travaux sont également développés par des amateurs ou par des industriels sans donner lieu à un article scientifique. Il est aussi possible de générer des reliefs qui ne correspondent pas forcément à des terrains (*e. g.* des paysages lunaires, des terrains extra-terrestres ou encore la surface d'astéroïdes). D'autres travaux cherchent à modéliser des parties très particulières d'un terrain comme les grottes et les caves.

Un des plus vieux algorithmes de génération procédurale de terrains consiste à ce qu'une particule dépose des sédiments selon une trajectoire aléatoire [117]. Si le déplacement de cet agent respect un chemin aléatoire (*random walk*), on génère alors un terrain fBm. Il est possible d'étendre cet algorithme avec des trajectoires plus complexes ou un système de stabilisation qui fait que les sédiments déposés vont glisser sur les pentes et remplir les minima locaux. Un autre algorithme *ad hoc* consiste à placer des demi-sphères sur un terrain, ce qui peut correspondre à un algorithme de *faulting* [1].

Plusieurs techniques cherchent à recréer des astéroïdes ou des paysages extraterrestres. MARTIN *et al.* [230] proposent plusieurs techniques de reconstruction d'astéroïdes. Ils cherchent également à reconstruire géométriquement des cratères à partir de données satellites [229] en extrapolant localement les détails à l'aide de bruit fractal : leur méthode permet de reconstruire des paysages lunaires qui ne sont pas idéalisés (et donc qui sont utilisables pour des simulations). JALOBEANU *et al.* cherchent un modèle complet de génération de relief, qui gère également des paramètres d'éclairage et le réglage d'une caméra virtuelle pour pouvoir simuler une photo prise par satellite. La génération du terrain se fait à l'aide de fonctions ondelettes sur des maillages surfaciques avec une géométrie arbitraire (au lieu d'une synthèse de Fourier dans le plan ou d'une composition d'harmoniques sphériques).

Dans le but de créer des terrains qui soient animés, VAN ECK et LAMERS [387, 388] observent des boîtes de Petri contenant différentes colonies de bactéries. À l'aide d'une caméra et d'une analyse des images, ils reconstruisent une carte de hauteurs qui se déforment au cours du temps.

GRELSSON [150] propose de générer le terrain avec des algorithmes connus de synthèse par bruits mais dans un ensemble de tuiles qui sont ensuite combinées à l'aide de différentes fonc-

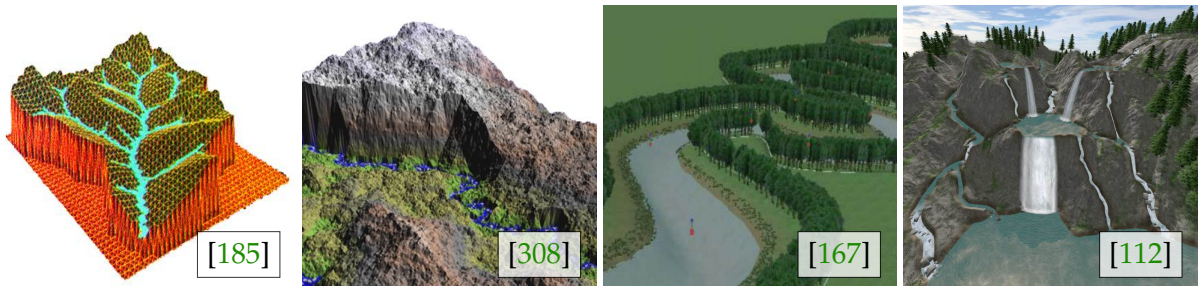


FIGURE 28 : Les rivières sont des éléments complexes à modéliser et à intégrer dans une scène naturelle. Des techniques de génération ou d'édition semi-automatiques permettent de créer facilement des méandres ou des cascades.

tions d'interpolation. Ces tuiles sont contraintes pour permettre à des morceaux de rivières ou de chemins d'être connectés avec leurs parcelles voisines.

PATEL propose un algorithme de génération d'îles se basant sur un ensemble de polygones [279]. Mêlant calcul de l'élévation, raffinement des polygones, définition des cours d'eau et des lacs internes, mise en place de biomes, son algorithme (qui s'inspire beaucoup des terrains conçus pour les jeux-vidéos) produit des reliefs très intéressants même si ils sont souvent similaires. S'inspirant des travaux de PATEL, GAINEY s'interesse aux techniques de génération procédurales de planètes [134].

2.3.2 Modélisation de rivières

Les rivières sont un des éléments les plus structurants du terrain. Plusieurs travaux se sont intéressés à la modélisation des rivières qu'elle soit automatique ou guidée par un utilisateur (Fig. 28). Nous excluons de ce chapitre les simulations d'écoulement de fluides qui sont présentées dans la section suivante.

KELLEY *et al.* [185] proposent en 1988 le premier algorithme de génération procédurale de bassins-versants. Bien que cet algorithme ne modélise qu'un seul bassin-versant, les résultats graphiques sont visuellement convaincants. La méthode fonctionne en dessinant les contours d'un bassin-versant et le canal associé ; progressivement ces deux entités vont être subdivisées jusqu'à obtenir le réseau hydrographique final. Finalement, une modification de l'algorithme du *midpoint displacement* permet de reconstruire l'élévation de tout le bassin-versant.

L'approche de KELLEY *et al.* était novatrice sur un point essentiel : les auteurs proposaient de créer la structure du réseau hydrographique avant le relief en tant que tel. Le processus à l'envers permet d'éviter de simuler les écoulements. Plusieurs travaux ont repris ces idées. En 2005, BELHADJ et AUDIBERT [24, 25] proposent d'utiliser des particules qui suivent un mouvement brownien pour tracer des crêtes et des rivières. En 2008, TEOH [375, 376] combine plusieurs algorithmes *ad hoc* pour dessiner des rivières, des méandres ou encore des deltas.

Certains travaux se sont concentrés sur la génération automatique d'une rivière, indépendamment de la notion de réseaux hydrographique. PRUSINKIEWICZ et HAMMEL [308] ont étudié ce pro-

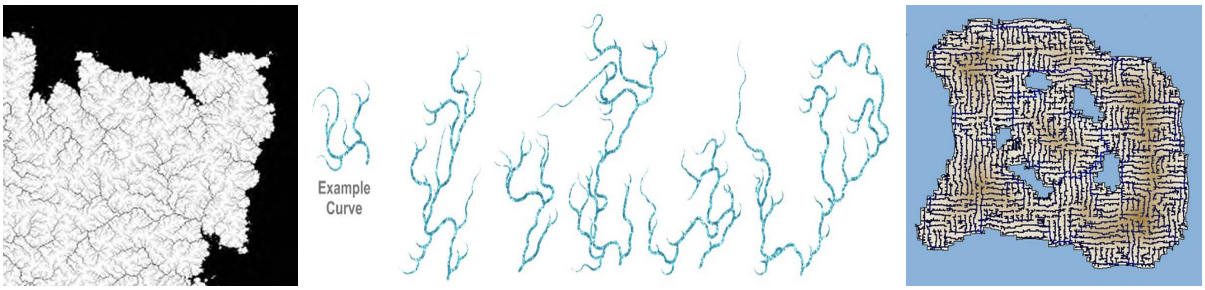


FIGURE 29 : Bien que les réseaux hydrographiques structurent le relief en créant des vallées, peu de travaux se sont intéressés à leur génération à l'échelle régionale.

blème et cherchent à tracer des trajectoires complexes de rivières sur un terrain fractal. Leur méthode fonctionne mais le relief créé présente des falaises exagérément élevées ; leur algorithme ne gère pas non plus la notion de tributaires (plusieurs cours d'eau qui se jettent les uns dans les autres) et la rivière créée possède une altitude fixe. KURWOSKI [194, 195] va plus loin en proposant de générer procéduralement des trajectoires de rivières qui peuvent tracer des méandres et des bras morts. Il ajoutera par la suite une méthode de construction [196] de plusieurs types de deltas. DE CARLI *et al.* [94] s'intéressent, eux, à la création de canyon, qu'ils forment en appliquant plusieurs filtres sur une carte de hauteurs segmentée. D'autres algorithmes proposés par des passionnés se sont intéressés à déformer le terrain le long d'une trajectoire calculée par des algorithmes de plus court chemin [74].

En 2015, PYTEL et MANN [312] proposent un système de génération de rivières souterraines. S'appuyant sur des travaux en hydromorphologie, ils construisent des réseaux géométriquement complexes grâce à un modèle de calcul de flot et de pression sur une grille voxélique.

Plusieurs articles se sont penchés sur le contrôle proposé à l'utilisateur. En concevant un système d'édition manuel de profils de rivières, HUIJSER *et al.* [167] permettent à un infographiste de construire manuellement et précisément la forme de la rivière ainsi que sa composition et le placement de la végétation. Les cascades, qui sont plus complexes à générer procéduralement car elles sont souvent non-planaires, ont été étudiées par EMILIEN *et al.* [110, 111, 112]. Leur approche est semi-automatique et l'utilisateur contrôle les trajectoires des cours d'eau. Ces travaux, postérieurs à la publication de notre article sur la génération de rivières, proposent une classification des chutes d'eau qui s'inspire de la segmentation des cours d'eau que nous avons utilisée.

On peut constater que peu de travaux se sont intéressés à générer des réseaux hydrographiques à grande échelle qui soient cohérents (Fig. 29). Un premier travail proposé par BARDEEN [18] et intégré dans MOJOWORLD semble créer le terrain et un réseau hydrographique en parallèle. Si les résultats sont visuellement impressionnants, aucune information relative à son implémentation n'est disponible. Seuls quelques articles exploreront cette approche : en particulier DERZAPF *et al.* [98]. Adaptant un algorithme de subdivision de terrain, ils génèrent de véritables réseaux hydrographiques à l'échelle planétaire. Les cours d'eau, qui se raffinent alors que la caméra se rapproche du sol, sont cohérents et se jettent tous dans la mer. SAPOZHNIKOV et NIKORA [337] créent des réseaux hydrographiques avec des chemins browniens. À partir

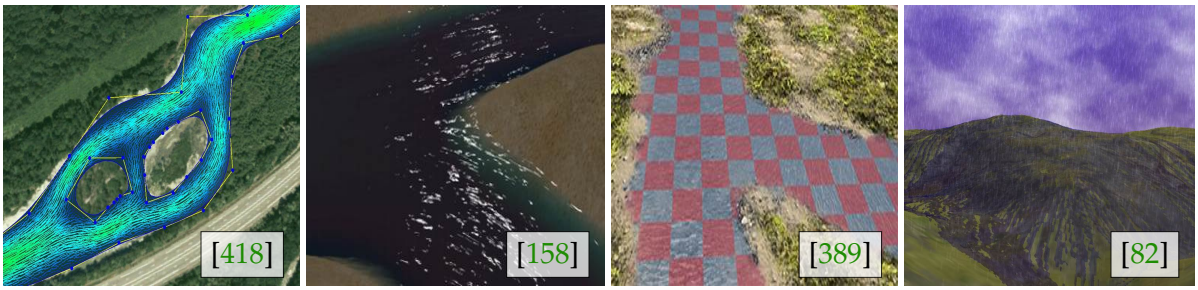


FIGURE 30 : Plusieurs techniques récentes se sont intéressées à animer des cours d'eau sans faire appel à des simulations coûteuses en temps de calcul.

d'exemples, MERRELL et MANOCHA [240] proposent de construire des réseaux hydrographiques. Les rivières sont approximées par leur trajectoires (*i. e.* des courbes 2D vectorielles) et l'intégration de cette technique pour générer un terrain avec du relief mériterait d'être approfondie.

Plusieurs travaux ont pour objet d'extraire les rivières d'une carte de hauteurs. L'algorithme le plus connu est D8 [269] proposé par O'CALLAGHAN et MARK. D'autres méthodes permettent de décomposer l'image en sous-régions et génèrent des bornes supérieures et inférieures de la squelettisation ce qui permet d'extraire un ou plusieurs ensembles connexes [16]. Ces algorithmes de calculs d'écoulements sur des cartes de hauteurs sont souvent utilisés dans les jeux-vidéos qui fonctionnent avec des mondes décomposés en tuiles [285].

Notre algorithme de génération procédurale de terrain [139, 140], présenté en Chapitre 5, s'inscrit donc dans ce cadre en proposant de générer des réseaux hydrographiques (à l'aide d'une grammaire de développement) avant de construire le relief du terrain. Plusieurs extensions de nos travaux – non publiées à ce jour – ont été étudiées dans le cadre de stages. VIMONT [395] propose de générer un réseau cohérent et multi-échelles en subdivisant et en raffinant des bassins-versants. Si les résultats en terme de structures sont intéressants, la méthode peut générer des auto-intersections lors du raffinement. GAILLARD [131] simplifie ces travaux en travaillant sur une grille régulière ce qui élimine les problèmes géométriques mais introduit des artefacts graphiques le long des directions principales de la grille, qui sont visuellement problématiques.

Enfin, plusieurs travaux concernent l'animation ou à l'écoulement des rivières (Fig. 30). L'animation de l'eau peut se faire par des méthodes classiques de simulations de liquides mais ces techniques sont coûteuses en temps de calculs. Des techniques de simulations spécialisées pour l'écoulement d'eau en temps-réel dans des environnement planaires existent et permettent d'extraire une surface d'eau en mouvement [188]. Pour plus d'informations sur la simulation d'écoulements liquides sur des surfaces solides, nous conseillons au lecteur de se tourner vers la thèse de EL HAJJAR [108]. NEYRET et PRAIZELIN ont proposé une méthode phénoménologique d'écoulement de petits cours d'eau [263]. Ces travaux seront étendus quelques années plus tard avec l'intégration des ondes de choc qui sont créées par des obstacles émergeant de l'eau [324]. Une autre technique pour animer la surface de rivières, capable de passer à l'échelle grâce à une description vectorielle des flux, est proposée par YU *et al.* en 2009 [418]. En 2010, une présentation de VLACHOS détaille comment l'écoulement de l'eau est géré dans les jeux-vidéos développés par VALVE [399]. Un système complet de génération et de rendu de rivière est également présenté

par Burrell [61]. Enfin, il existe plusieurs techniques basées sur la décomposition de la rivière en un ensemble de petits carreaux chacun animés [70, 158, 389].

D'autres phénomènes d'écoulements sont très souvent ignorés mais peuvent être très importants visuellement dans l'animation d'un cours d'eau. Par exemple, Coutinho *et al.* [82] propose de gérer l'écoulement de l'eau de pluie sur le terrain (ce qui peut engendrer un débordement de rivière). L'eau peut également rencontrer différents matériaux dans le lit d'une rivière, comme le sable. Rungjiratananon *et al.* [330] montrent une méthode d'écoulement qui tient compte de ces interactions eau/sable qui peuvent avoir un impact sur le mouvement et le rendu de l'animation.

2.3.3 Simulations

Un terrain réel est le fruit de plusieurs millions d'années d'évolution naturelle. Les processus qui sculptent un paysage sont nombreux et agissent à plusieurs échelles. La tectonique des plaques est responsable de la création des chaînes montagneuses et des océans. Les glaciers creusent les vallées alors que l'érosion hydrique forme les rivières. Le vent et le sable sont responsables de la forme de certains monticules rocheux. Les réactions chimiques (oxydation, corrosion) changent la couleur et l'apparence des pierres.

Il est possible de simuler ces processus sur des données digitales. Ces algorithmes ne génèrent pas des terrains mais les transforment. On a donc toujours besoin d'un point de départ (habituellement un terrain généré par une méthode fractale). Ces processus ne peuvent pas être simulés tous en parallèle et très peu de travaux ont essayé d'unifier ces transformations. De la même façon, ces algorithmes ont souvent été conçus pour manipuler des cartes de hauteurs (cette structure est simple d'utilisation, elle est planaire et régulière), même si certaines simulations ont été adaptées pour tourner avec des voxels ou des TINs.

2.3.3.1 Stabilisation et tectonique

La tectonique des plaques est un des facteurs déterminants dans la création des chaînes de montagnes et des fosses océaniques. La Terre peut être vue comme un ensemble de plaques (continentales ou océaniques) qui sont chacune en mouvement. Ce mouvement est créé par l'apport de matières au niveau des dorsales océaniques (et parfois continentales).

La tectonique des plaques n'a pas fait l'objet de beaucoup de travaux de modélisation en informatique graphique (Fig. 31). Les algorithmes de *faulting*, qui correspondent aux premières techniques de génération procédurales de terrains ne reposent sur aucune simulation physique. Jarocha-Ernst [175] teste le chevauchement de deux plaques l'une sur l'autre ; c'est l'un des premiers travaux qui simule réellement le mouvement des plaques tectoniques même si les résultats sont décevants, d'un point de vue graphique. Un modèle similaire sera proposé par Viitanen [394] qui l'étendra à la gestion de plusieurs plaques. Chaque plaque tectonique va se déplacer, et suivant son type (océanique ou continentale) va plonger ou s'écraser lors d'une collision. Les résultats produits sont beaucoup plus réalistes et on voit même apparaître des chaînes de montagnes, des archipels, *etc*.

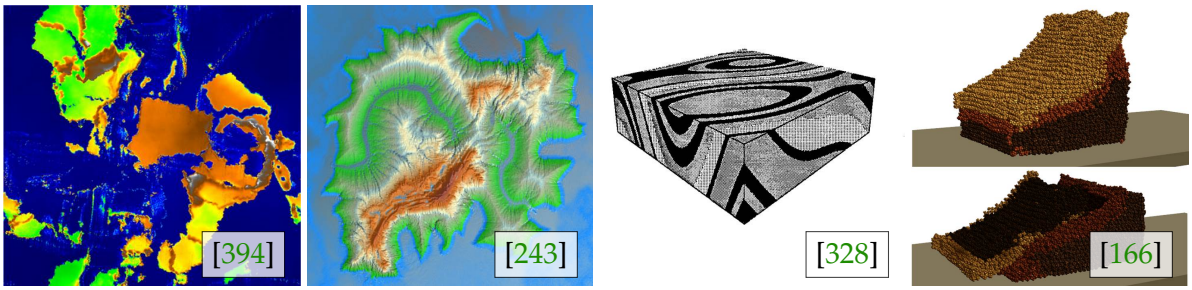


FIGURE 31 : Peu de travaux ont pour objet la simulation des sols. La tectonique est pourtant une des causes de la formation des chaînes montagneuses. La géologie interne d'un terrain modifie l'effet de l'érosion hydrique et peut même causer des effondrements.

Récemment, MICHEL *et al.* [242, 243] ont ajouté du contrôle en proposant à l'utilisateur d'esquisser rapidement un contour d'île, quelques rivières et des chaînes de montagnes. Une fois ces informations dessinées, son algorithme va générer des plaques tectoniques dont la forme respecte l'esquisse. Il rajoute ensuite une structuration hydrologique. Les vallées créées par les rivières vont générer du terrain relativement lisse alors que les zones de collisions entre les plaques vont faire apparaître des régions montagneuses.

À une échelle plus petite, ROUDIER *et al.* [328] proposent de modéliser un terrain à l'aide de différentes couches géologiques et de simuler des plis géométriques dans le terrain afin de complexifier sa structure interne. Chaque couche ayant des propriétés différentes en matière d'absorption, de saturation et de solvabilité, elles réagissent toutes différemment sous l'influence de l'érosion. HUDÁK et ĎURIKOVIČ [166] simulent des effondrements et des éboulements de terrains à l'aide de particules. Avec une méthode différente et adaptée aux structures volumiques, ITO *et al.* [173] créent des parois rocheuses réalistes.

Les forces de gravité et de friction engendrent également un phénomène appelé stabilisation des matériaux. Tout relief, composé de particules plus ou moins grosses (comme le sable, les graviers ou les rochers) possède un angle de repos. Toute particule qui tombera sur un tas au repos avec cet angle ne subira pas une force de frottement suffisante pour se maintenir sur la pente (Fig. 32). MUSGRAVE *et al.* proposent de simuler cet effet de stabilisation [255] ; il parle alors

FIGURE 32 : La stabilisation (également souvent nommée érosion thermique (*thermal erosion*) est responsable de la formation de tas et de pentes douces.

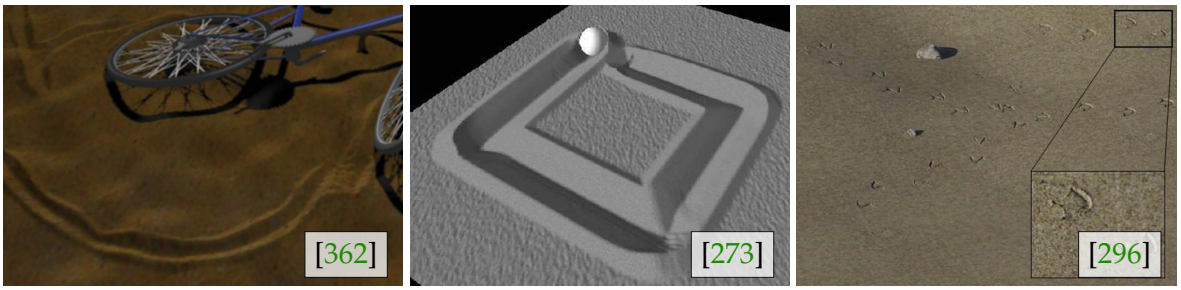


FIGURE 33 : Les travaux sur la représentation des traces découlent directement du phénomène de stabilisation. Ces traces sont très utilisées dans les jeux-vidéo et les films pour leur capacité à rendre les scènes plus vivantes.

d'érosion thermique[19] (*thermal erosion*). Sur une carte de hauteurs, il détecte les pentes trop fortes et distribue la matière pour égaliser le relief. MARÁK *et al.* [228] proposent un système de réécritures de terrains (sous forme d'une grammaire à appliquer sur une carte d'élévations) censé être capable de généraliser plusieurs types d'érosion, dont la stabilisation.

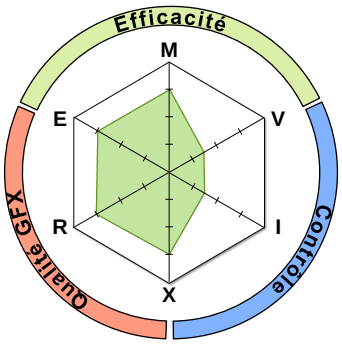
BENEŠ *et al.* [30] adaptent la stabilisation à l'aide d'une structure informatique plus complexe de piles de matières puis la rendent facilement parallélisable [31]. En 2005, ils présentent [29] un processus similaire spécifiquement conçu pour simuler la création de mesas. OLSEN [270] utilise une version légèrement modifiée de cette érosion pour créer des terrains utilisables dans des jeux-vidéos (il parle de terrains *applicable*). JAKO et TÓTH [178] proposent un modèle GPU de stabilisation couplé avec de l'érosion hydrique. PYTEL et MANN [310, 311] décident d'utiliser un modèle de calcul de répercussions en cascade (*avalanching*) et y adaptent l'algorithme de stabilisation.

La simulation de la stabilisation n'est pas seulement un phénomène à l'échelle des années ou des siècles ; elle peut être utilisée pour la création de traces au sol (Fig. 33). La moteurs de rendu temps-réel ne déforment que virtuellement la géométrie à travers des *normal maps*.

SUMNER *et al.* [361, 362] montrent qu'il est possible d'utiliser ce type d'érosion pour modéliser des traces (de pas, de pneu, ...) dans différents types de sols (boue, sable, neige). ONOUE et NISHITA [273] cherchent à modéliser les traces et les interactions entre des objets et le sable. Cette approche sera reprise par PEYRAT *et al.* [296] pour modéliser toute trace (d'animaux, de véhicules ou d'humains) par déformation/raffinement local des maillages en utilisant les propriétés de voisinage des G-Cartes.

Enfin, des simulations plus réalistes et plus précises existent : par exemple, LI et MOSHELL [217] modélisent le déplacement de la terre de manière plus complexe, ce qui permet par exemple d'imiter l'effet d'un bulldozer sur de la terre. CHANCLOU *et al.* [69] modélisent les sols meubles avec un système de particules.

19. Ce terme semble faux. La *thermal erosion* est définie comme l'érosion du permafrost causée par les actions thermiques et mécaniques de l'eau. Le terme de *thermal weathering* décrit l'action de la chaleur sur les roches (et leur effritement). Si MUSGRAVE n'utilise plus les mots *thermal erosion* par la suite (il parle plus généralement de « mouvements de terrains »), ce terme est resté dans la littérature de l'informatique graphique.

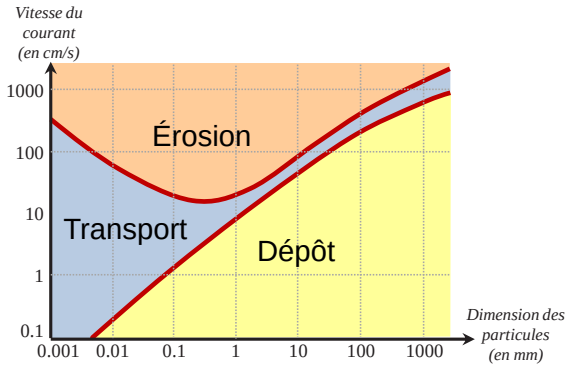


Analyse : plusieurs algorithmes simulant le mouvement du sol existent et permettent de reproduire la plupart des phénomènes réels quelle que soit leur échelle (stabilisation de l'ordre du cm, effondrement de l'ordre du mètre, tectonique des plaques de l'ordre de la dizaine de km). Malgré tout, la nature très différente de ces algorithmes fait qu'il est **difficile d'unifier leurs différents effets sur le relief** au sein d'un unique *framework*. C'est probablement la raison principale pour laquelle ces méthodes sont encore peu utilisées dans des logiciels professionnels. De plus, ces algorithmes sont d'autant plus adaptés qu'ils s'appliquent à des terrains volumiques (ou du moins dont la composition du sol est hétérogène) ce qui permet d'obtenir des effets complexes d'effondrements ; malheureusement, peu de logiciels proposent de manipuler ce type de représentation.

2.3.3.2 Érosion hydrique

Le cycle naturel de l'eau, présent tout au long de l'année, érode tous les paysages et conditionne leur structure. L'eau agit de plusieurs manières : quand elle s'écoule le long des pentes, qu'elle s'évapore ou s'infiltre dans les sols, quand elle humidifie les roches et change leur température, quand elle change de forme pour devenir de la neige ou de la glace ou encore quand elle s'écrase sous forme de vagues sur les côtes. L'érosion hydrique est un des facteurs les plus importants dans la création des reliefs terrestres.

L'érosion hydrique est provoquée par le mouvement de l'eau sur un terrain. Ce mouvement, quand il est suffisamment rapide, va arracher au relief de petits morceaux sédimentaires qui seront transportés en aval. Une fois que le courant devient plus faible, les particules sédimentaires se re-déposent par gravité sur le fond du lit de la rivière. L'ensemble de ce processus est décrit par le diagramme de Hjulström qui indique l'état et l'activité d'un sédiment en fonction de sa taille et de la vitesse du cours d'eau.



Pseudo-simulations

De nombreuses études se sont intéressées au processus d'écoulement de l'eau sans forcément recourir à des simulations réalistes (Fig. 34). Pour modifier ses paysages fractals et leur donner une structure, Musgrave *et al.* [255] utilisent deux types d'érosion : la stabilisation, et un calcul *ad-hoc* de l'érosion hydrique sur des cartes de hauteurs. Comme pour la stabilisation, Olsen [270] optimise l'érosion hydrique proposée par Musgrave *et al.* et cherche à créer des terrains utilisables dans le cadre de jeux-vidéos. Roudier *et al.* [328] enrichissent le modèle d'érosion de Musgrave *et al.*en intégrant les notions d'infiltration de l'eau et de dissolution chimique qui

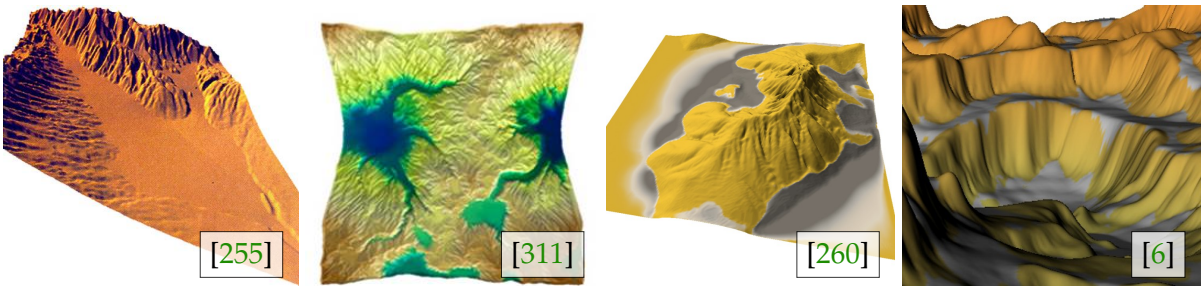


FIGURE 34 : De nombreuses méthodes *ad hoc* fonctionnant sur des cartes d'élévation ou avec des particules newtoniennes imitent les effets provoqués par l'érosion hydrique sans pour autant s'appuyer sur une simulation physique des causes réelles.

agissent sur plusieurs couches géologiques. NAGASIHMA [256] étudie lui aussi le rôle combiné de la stabilisation et de l'érosion hydrique. D'AMBROSIO *et al.* [91] utilisent un simple automate cellulaire pour simuler le déplacement de l'eau et parviennent à obtenir des résultats réalistes.

Ce type d'approche qui travaille directement sur la carte de hauteurs, est généralisé par PYTEL et MANN [310, 311]. Ils proposent d'utiliser une approche en cascade (*avalanching*) qui correspond à un méta-algorithme où une transformation effectuée sur la carte de hauteurs augmente les chances que ce même changement soit effectué dans le voisinage. Ils adaptent avec cette approche en cascade, l'algorithme de stabilisation et l'érosion hydrique.

BENEŠ et FORSBACH [32] clarifient la notion d'érosion hydrique et montrent qu'elle peut être décomposée en quatre étapes : l'apparition d'eau (*water's apparition*), la récupération de la matière (*water's erosion*), le transport le long des pentes (*water's transportation*), et l'évaporation (*water's deposition*) ; ces étapes peuvent être paramétrées et étudiées indépendamment les unes des autres. Ils proposent aussi d'utiliser une valeur limite de saturation de l'eau, au delà de laquelle les sédiments en suspension se déposeraient.

CHIBA *et al.* [72], dans une approche Eulérienne 2D mixée avec de la physique Newtonienne, utilisent des particules qui ont une vitesse et donc de l'inertie : l'eau ne s'écoule plus selon une vitesse uniforme sur la carte de hauteurs. SUTHERLAND et KEYSER [363] étendront ce modèle pour y ajouter un mécanisme de retenue (*pooling*) afin de créer de véritables maillages d'eau. NEIDHOLD *et al.* [260] parviennent à optimiser suffisamment ce modèle newtonien pour contrôler de manière interactive le placement des sources d'eau et les paramètres d'érosion. Bien que cette approche ne soit pas facilement parallélisable, ANH *et al.* [6] arrivent à porter ces calculs sur le GPU, pour éroder des terrains de grande taille.

Approches Eulériennes 3D (NSE)

En 2006, BENEŠ [34] propose la première vraie simulation de liquide pour éroder des terrains. Il résout les équations de Navier-Stokes (NSE) selon une approche Eulérienne ; même si la grille 3D est de taille et de résolution limitées, il lui est possible de simuler le comportement volumique de l'eau (Fig. 35). Couplée à une représentation multi-matériaux voxélique de l'environnement, sa simulation peut éroder le terrain de façon beaucoup plus réaliste. WOJTAN *et al.* [411]

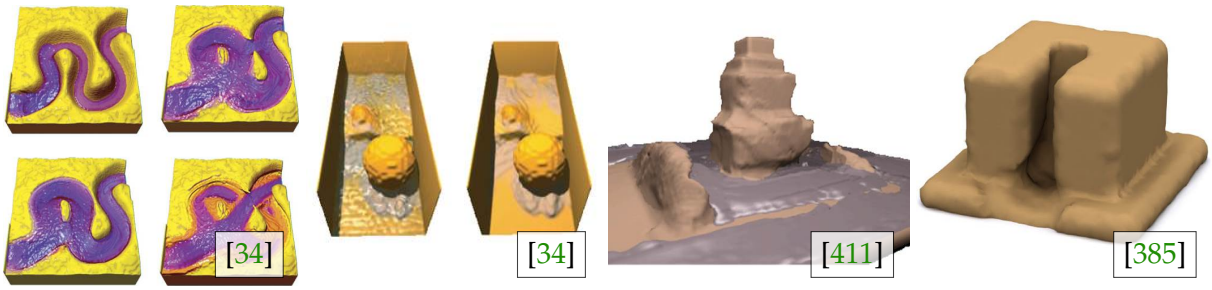


FIGURE 35 : Les simulations physiques de l'eau à l'aide de schémas de résolution Eulériens 3D des équations de Navier-Stokes permettent d'obtenir un comportement réaliste des liquides.

généralisent ce type de simulations à d'autres liquides ce qui leur permet de modéliser d'autres formes d'érosion plus complexes comme la corrosion. Un processus d'érosion hydrique 3D est adapté pour des structures non voxéliques comme les maillages de Delaunay déformables par TYCHONIEVICH et JONES [385].

Approches Eulériennes 2D (SWE)

Les équations de Navier-Stokes étant complexes à simuler, il est possible de recourir aux équations de Saint Venant (*shallow water equations* ou SWE) qui sont beaucoup plus simples à résoudre. Ces équations qui manipulent des colonnes d'eau, ignorent les effets verticaux de déplacement de l'eau à l'intérieur de ces colonnes (Fig. 36). En 2007, BENEŠ [28] et MEI *et al.* [239] proposent tous les deux des systèmes d'érosion qui reposent sur ce modèle. Les résultats sont réalistes, et beaucoup plus rapides à calculer qu'avec Navier-Stokes mais ne sont pas applicables sur des terrains non-planaires. Ils montrent également que les calculs sont facilement parallélisables. Un an plus tard, ŠŤAVA *et al.* [365] appliquent cet algorithme sur GPU en découpant le terrain en différentes tuiles qui sont gérées chacune indépendamment. Ce système sera étendu par VANEK *et al.* [392] qui l'appliqueront sur des données de plusieurs dizaines de Go. JAKO et TÓTH [178] proposeront également une approche GPU du même modèle, pour simuler à la fois l'érosion hydrique SWE et la stabilisation des matériaux.

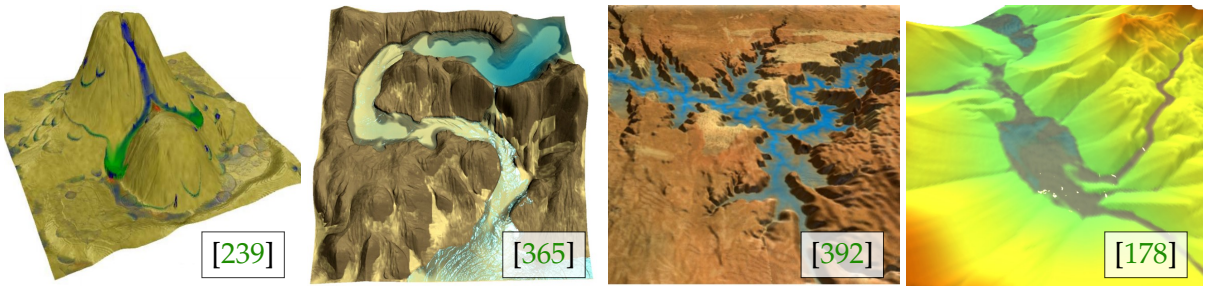


FIGURE 36 : L'utilisation de modèles dits *shallow waters* permet de négliger les mouvements verticaux et de simplifier les calculs d'écoulements.

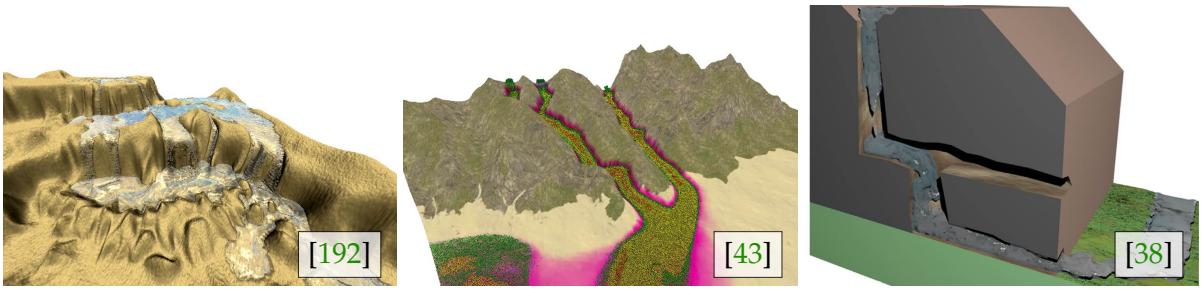
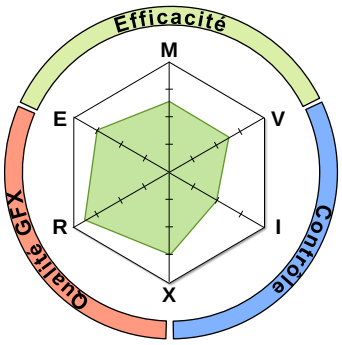


FIGURE 37 : L'utilisation de SPH ou des particules Newtoniennes permet de modéliser intuitivement des écoulements. Le mouvement de ces particules s'adapte facilement à des scènes géométriques complexes tout en restant simple à calculer sur GPU.

Approches Lagrangienne 3D

Les SPH (*smoothed particle hydrodynamics*) sont une approche de résolution Lagrangienne utilisable pour des problèmes complexes comme la résolution des équations de Navier-Stockes. Ce cadre de résolution fait intervenir des particules en mouvement et a été appliqué dans le cadre de l'érosion hydrique (Fig. 37). KRIŠTOF *et al.* [192] proposent une approche mixte entre résolution Lagrangienne 3D avec des particules SPH et résolution Eulérienne avec des particules fixes sur le relief. Ces deux types de particules échangent des matériaux lors de l'érosion ou de la sédimentation. Une méthode similaire est proposée par KUROWSKI [194] qui cherche à simuler la formation des méandres. Pour cela, il rajoute un processus *ad hoc* qui va attirer les particules chargées de sédiments vers les bords intérieurs de la rivière : progressivement, les parties convexes vont être érodées et les parties concaves sédimentées/remblayées. Il étendra ces travaux pour générer des bras morts [195] et des deltas [196]. En 2010, BÉZIN *et al.* [42, 43] travaillent sur une simulation de particules Newtoniennes 3D pour éroder des maillages planaires de très grande taille. En 2013, ils passent à un algorithme d'érosion [37, 38] à base de SPH qu'ils utilisent dans des environnements 3D (stockés sous la forme de 3G-cartes) ce qui leur permet de simuler, par exemple, la formation d'arches de mer.

L'ensemble des méthodes décrites simule le comportement de l'eau à la surface, et ne gère pas (ou alors de manière *ad hoc*) les notions d'infiltration des liquides. Récemment, ANDRYSCO *et al.* [5] ont proposé de coupler une simulation Lagrangienne de fluides avec une représentation de la perméabilité, de la porosité et de la capillarité des matériaux. L'intégration de ce type d'approche réellement volumique pourrait changer la manière dont l'érosion se produit.



Analyse : les simulations d'érosion hydriques **étendent naturellement les travaux sur les calculs d'écoulements de liquides** qui sont utilisés en informatique graphique depuis plusieurs années. En ajoutant des notions de modification du relief (l'érosion, le transport, le dépôt de sédiments mais aussi l'infiltration ou l'évaporation du liquide), il est possible de transformer des paysages fractals pour les rendre plus réalistes. Les simulations d'érosion sont aujourd'hui suffisamment rapides à calculer pour permettre leur incorporation dans des outils de modélisation interactive (cf. la démo WebGL [47]). Par contre, la plupart de ces méthodes ne permettent pas à proprement parler d'extraire une animation d'eau qui s'écoule de manière stable (dans le cas où l'on souhaite par exemple animer une rivière).

2.3.3.3 Phénomènes éoliens

Le vent façonne les paysages de plusieurs façons. Il agit, par exemple, sur la végétation qui peuple le relief et peut déformer les arbres pendant leur croissance [301]. L'érosion éolienne (*aeolian erosion*) est également responsable de la forme de certains rochers.

Le vent peut également agir directement sur le relief de surface, quand celle-ci est composée de petites particules comme le sable. Plusieurs études se sont intéressées à la formation de paysages désertiques, résultat d'une dynamique complexe [204]. Grâce à des observations en conditions réelles ou artificielles [4], les chercheurs ont pu établir un certain nombre d'équations qui régissent ces phénomènes [241, 264].

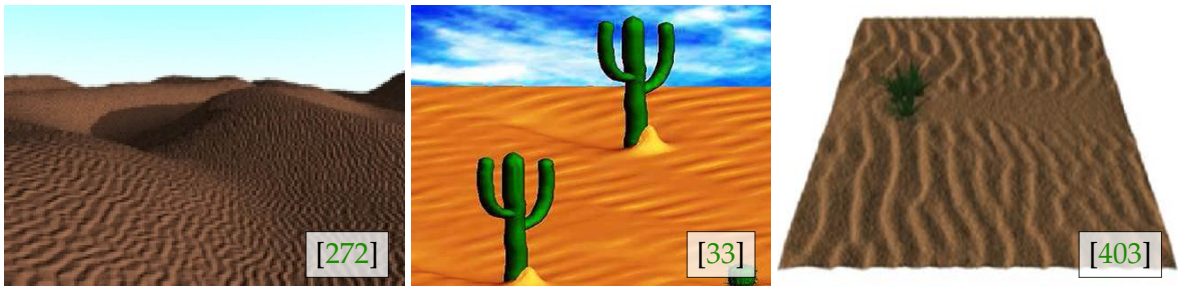
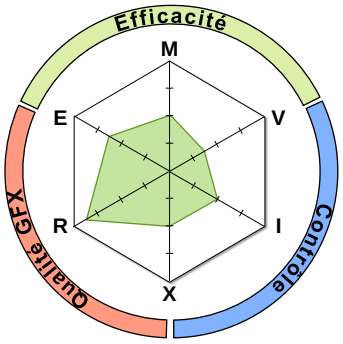


FIGURE 38 : Les simulations de zones désertiques permettent de recréer des paysages où le sable agit à plusieurs échelles en formant, ou des draas (collines de sables), ou des dunes, ou des ridules.

En informatique graphique, on simplifie généralement ces simulations (Fig. 38). En 2000, ONOUE et NISHITA [272] rappellent les trois composantes du mouvement des particules : la saltation (le déplacement par une série de sauts des particules), la reptation (le déplacement au sol des particules les plus lourdes) et la suspension (le déplacement en vol des particules les plus légères). Ils décrivent ensuite les modèles de simulation des ridules de sables (*wind-ripple model)* et de formation des dunes. Un mécanisme de niveaux de détails permet de filtrer les ridules quand la caméra est loin du relief. Quelques années plus tard, BENEŠ et ROAT [33] étendent cet algorithme en intégrant des objets qui feront obstacles au vent. Le sable s'accumule alors devant ces paravents et, à l'inverse, les ridules n'apparaissent plus dans leur ombre.

Utilisant un système de particules, WANG et HU [403] proposent une simulation bien plus complexe et réaliste des scènes désertiques. Au sein d'un même *framework* sont gérés les effets de saltation, de reptation et de suspension. De la même façon, un modèle de végétation interagit avec les particules. L'ensemble de ce système est implémentable sur GPU ce qui permet une simulation temps-réel.



Analyse : les phénomènes éoliens sont **surtout étudiés dans le contexte de scènes désertiques** (pour la création de ridules de sable, de dunes, *etc*). À notre connaissance, aucun travaux ne s'est par intéressé à simuler les effets du vents sur des volumes (par exemple pour la sculpture des roches). Les simulations de vent sont souvent dictées par des champs de vecteurs relativement simples et il serait intéressant d'avoir des méthodes procédurales pour construire des écoulements éoliens à grande échelle (voire à de multiples échelles) ce qui permettrait d'influencer comment la végétation pousse et quels matériaux sont présents à la surface du terrain. La combinaison de ces phénomènes avec des paramètres de température ou d'humidité pourrait donner des résultats d'un intérêt certain sur le plan visuel.

2.3.3.4 Dépôt de neige et de glace

La neige, le froid et la glace ont des effets importants sur le relief en hiver (Fig. 39). L'érosion provoquée par les glaciers semble être un facteur primordial dans la création de certains types de vallées, notamment celles dont le profil est en U. Des simulations très coûteuses à destination des spécialistes (qu'ils soient géologues ou climatologues) existent [287] mais sont difficilement exploitables dans le cadre de l'informatique graphique.

Les avalanches, qui sont le résultat combiné de chutes de neige et de problèmes de stabilisation des plaques neigeuses, ont été étudiées par RICHARD [321], par LAIGLE *et al.* [200] et par TSUDA *et al.* [382]. Ces processus sont souvent reproduits à l'aide de simulations à base de particules de type SPH.

L'enneigement peut être simulé avec des *metaballs* [265], avec des particules représentant les différents flocons [116, 161, 251] ou avec des piles de matières [118, 119]. Ces simulations sont très coûteuses et ne permettent pas de recouvrir des paysages de grandes tailles.

FOLDES et BENEŠ [120] sont les seuls à proposer une pseudo-simulation capable de gérer de grands terrains. Se basant sur une procédure approximant l'occlusion locale, ils calculent la quantité de neige qui tombe en un point, en fonction de la dissipation thermique et de l'illumination directe. Les changements de température sont également responsables de l'enneigement et de l'apparition de couches de glaces sur les lacs ou les rivières. MARÉCHAL *et al.* [226, 227] simulent tous ces effets sur des scènes composées de plusieurs matériaux, en prenant en compte la diffusion de la chaleur, l'inertie thermique ou encore l'ensoleillement.

Une technique très différente, proposée par PREMOŽE *et al.* [306] cherche à s'appuyer sur des images panchromatiques (des images en noir et blanc) pour estimer la couverture neigeuse (et la

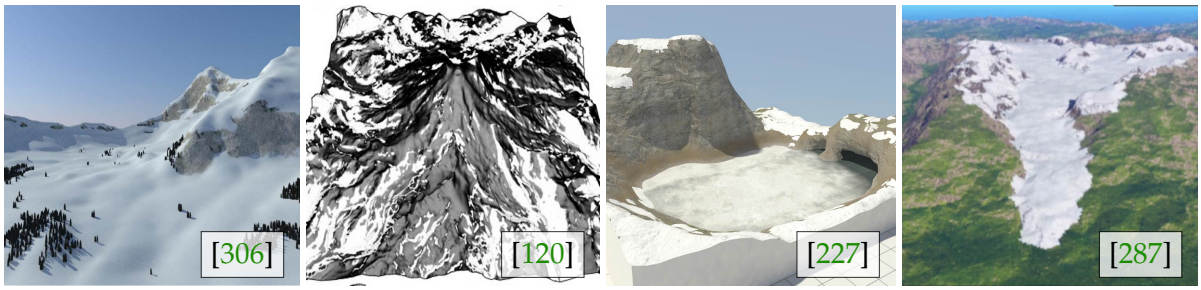
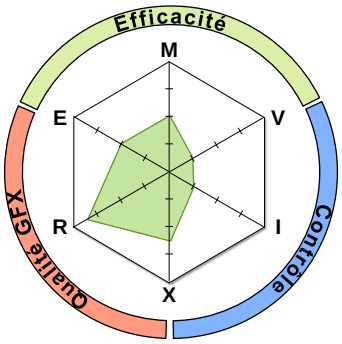


FIGURE 39 : Le froid, la neige et la glace ont des effets visuels déterminants sur les paysages montagneux. Ces phénomènes n'ont pas seulement un impact sur l'aspect (la texture, les matériaux) mais aussi sur la géométrie du relief, en particulier quand on les considère sur des périodes de temps longues.

présence de forêts et de roches) et permettre une reconstruction physiquement et visuellement plausibles de la scène.



Analyse : les simulations de neige sont presque **exclusivement dédiés aux petites scènes** à haute résolution (et souvent, ces scènes sont urbaines ou semi-urbaines). Peu de travaux se sont intéressés à proposer des modèles procéduraux de chutes de neige et de formation de glaces dans des environnements naturels ouverts et de grandes tailles. L'animation de l'apparition (et de la disparition) de la neige tout comme la transformation de son apparence (mouillée, sèche, propre, sale) n'ont jamais été étudiées dans des contextes d'applications temps-réel. Enfin, des phénomènes plus complexes (comme celui des avalanches ou de la formation des stalactites de glace) sont souvent traités dans des *frameworks* autonomes.

2.3.3.5 Autres érosions

D'autres phénomènes peuvent agir sur les paysages réels et en modifier le relief (Fig. 40).

Les érosions chimiques ne changent pas simplement l'apparence des roches ; elles peuvent avoir également un impact sur leur forme. DORSEY *et al.* [101] cherchent à vieillir des objets au contact de l'air et proposent d'utiliser une représentation avec une surcouche qui est progressivement dégradée (par un bombardement de particules, par la transmission de l'humidité vers les couches inférieures ou encore par la dissolution des minéraux).

Les chaos rocheux sont formés par plusieurs phénomènes de météorisation qui lissent progressivement les roches, dont l'altération sphéroïdale (*spheroidal weathering*). Utilisant une structure voxélique pour ses objets, BEARDALL *et al.* [21] proposent d'imiter ce phénomène en éliminant itérativement les parties saillantes d'une roche, c'est-à-dire les parties à forte courbure 3D. Avec ce système, ils arrivent à créer des formes de roches très particulières, appelés cheminées de fée ou hoodoo. TYCHONIEVICH et JONES [385] adaptent l'altération sphéroïdale à des maillages

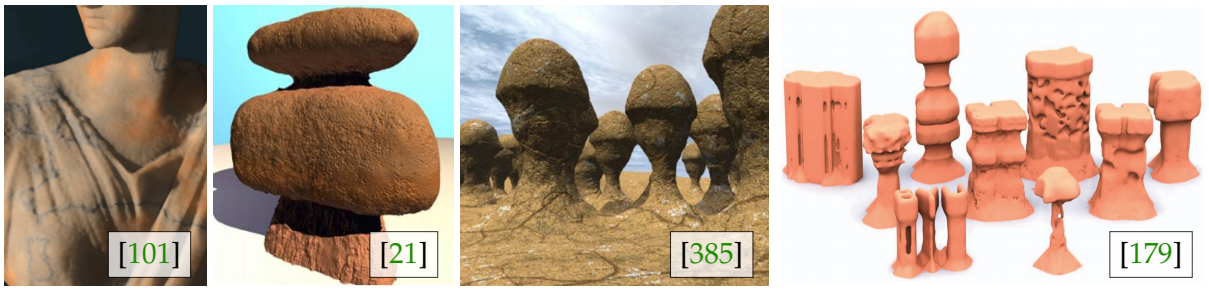


FIGURE 40 : Les érosions chimiques, physiques ou biologiques tendent à sculpter les roches apparentes. L'altération sphéroïdale, qui en fait partie de ces phénomène, est très simple à modéliser.

de Delaunay déformables. JONES *et al.* [179] intègrent à ce phénomène, une érosion chimique provoquée par la présence de sels (*cavernous weathering*) qui permet de créer des formes en creux arrondies appelées taffoni. En simulant tous les types de liquides ou de gaz dans un environnement 3D, WOJTAN *et al.* [411] arrivent à reproduire des effets d'érosion plus complexes comme la corrosion.

De nombreux phénomènes ne sont pas encore pris en compte en informatique graphique. Parmi les plus importants, on peut citer la cryoclastie, c'est-à-dire le changement de température (saisonnier, voir quotidien en haute montagne) qui crée un cycle de gel et dégel de l'eau, qui, *in fine* fait éclater les roches par dilatation au niveau des microsfissures. La thermoclastie peut agir sur la structure cristalline des minéraux. La desquamation et l'exfoliation sont des processus résultant de ces variations de température : la roche se dégrade et de petites écailles ou lamelles se détachent à la surface de la pierre. Des réactions chimiques, comme l'haloclastie en présence de sel, ou des phénomènes d'hydratation, d'oxydo-réduction, d'hydrolyse, de corrosion ou de carbonatation peuvent produire des résultats similaires. Sans aucune transformation chimique, une roche peut également se fracturer en interne à cause d'une variation trop importante de la température (on parle de choc thermique).

La façon la plus simple de casser une roche en plusieurs morceaux reste la fracturation à base d'impacts : cette érosion physique est très liée à l'instabilité gravitaire où des éboulements peuvent provoquer la création de nombreux petits rochers. Une étude récente [189] vient de mettre en lumière un nouvel agent d'érosion longtemps sous-estimé : la foudre. Chaque fois qu'elle frappe une surface dure, elle brise une dizaine de mètres cubes de roche. L'apport d'énergie est très important et la chaleur peut atteindre entre 8000 et 30000 degrés Celsius au point d'impact. Sachant qu'une zone montagneuse peut être frappée plusieurs dizaines de fois par km^2 par orage, les quantités de pierres brisées et éjectées sont suffisamment importantes pour expliquer, en partie du moins, l'aspect déchiqueté de ces zones montagneuses.

2.3.4 Contrôle de l'édition

Depuis une vingtaine d'années, beaucoup de travaux se sont intéressés au contrôle et à l'édition. Fournir aux infographistes des outils pour créer ou modifier un paysage de manière in-

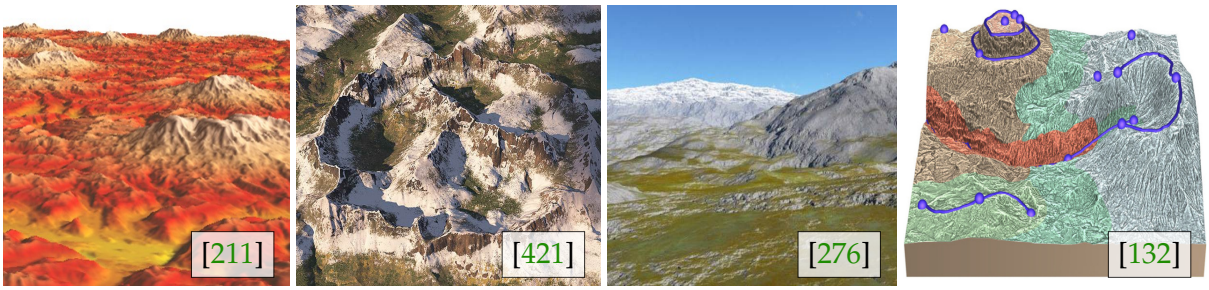


FIGURE 41 : La synthèse par l'exemple permet de créer facilement un paysage d'apparence similaire aux données d'entrée. Un contrôle plus fin agissant sur le placement ou la structure des reliefs reste toutefois soit difficile.

tuitive est devenu une priorité pour des questions de temps de développement et de coûts de production.

2.3.4.1 Construction par l'exemple

Ces dernières années, plusieurs travaux se sont appliqués à utiliser des données réelles pour synthétiser des terrains ; on parle de synthèse par l'exemple (*example-based synthesis*). De nombreuses techniques de génération procédurale de textures-couleurs existent [107, 135, 145, 146, 157, 211], tout comme existent plusieurs états de l'art récents sur la synthèse de textures 2D [210, 407] et 3D [300]. Malheureusement, peu de ces techniques sont utilisables pour générer les cartes de hauteurs correspondant au relief du terrain : le paysage qui est synthétisé n'offre pas de structure cohérente et son aspect reste très homogène [211]. Si quelques travaux ont essayé de corriger ces défauts [84, 85], il est souvent nécessaire de se tourner vers des techniques spécifiquement dédiées à la génération de paysages (Fig. 41).

Les travaux de CHIANG *et al.* [71, 383] sont parmi les premières techniques à utiliser des données réelles sous forme de cartes de hauteurs pour synthétiser un terrain. L'utilisateur pose des structures 3D en forme de prismes sur un terrain : chaque prisme correspond à une partie d'une chaîne de montagnes et chaque face triangulaire correspond à un profil transversal du relief. L'algorithme recherche ensuite dans une base de données des sous-parties d'images qui peuvent correspondre (à une déformation près) à ces prismes.

BROSZ *et al.* [59] s'intéressent à capturer les détails d'un terrain réel pour les reproduire sur un autre paysage. Si les hautes fréquences extraites sont replacées intelligemment sur des zones similaires, leur approche ne permet pas d'extraire des grands morceaux de reliefs.

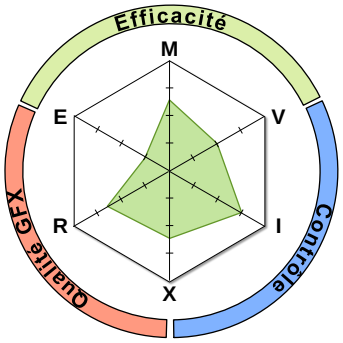
DACHSBACHER *et al.* [88, 89] proposent d'adapter ces techniques en prenant en compte les différentes dérivées du relief dans le calcul de similarité. Malgré la simplicité de l'approche, les résultats sont très prometteurs et permettent de combler des morceaux manquants du terrain.

La technique la plus citée quand on parle de synthèse par l'exemple est la méthode de ZHOU *et al.* [421]. Ils proposent à l'utilisateur de dessiner quelques éléments marquants du relief qu'il souhaite (des crêtes et des vallées). En analysant une base de données de terrains réels, ils

cherchent à retrouver des morceaux de terrains qui pourraient remplacer (moyennant une déformation) l'esquisse de l'utilisateur.

Cette technique de synthèse a été utilisée dans plusieurs travaux où l'utilisateur édite un terrain. Si TASSE *et al.* [369, 372] permet à l'utilisateur d'esquisser les crêtes les plus importantes de son paysage de manière grossière, elles sont ensuite remplacées par des morceaux de terrains réels ayant la même structure et qui viennent d'une base de données. Le reste du terrain est, lui, généré procéduralement pour ne pas avoir de reliefs marquants. DOS PASSOS *et al.* [102] analysent un ensemble de morceaux de terrains selon plusieurs points de vue pour en extraire des silhouettes ; elles sont utilisées pour trouver des reliefs qui correspondront aux esquisses dessinées par l'utilisateur et les remplaceront. Dans le système proposé par GAIN *et al.* [132], la synthèse par l'exemple peut être contrôlé par différentes contraintes qui sont esquissées.

Une autre approche possible est d'analyser des terrains existants et d'utiliser certaines caractéristiques statistiques pour générer des reliefs similaires. PARBERRY propose [276] de générer des terrains à l'aide d'un bruit à valeurs et d'ajuster les élévations pour correspondre à un histogramme d'altitudes réelles. Il fait de même [277, 278] avec un bruit de Perlin et des distributions de gradients (selon plusieurs échelles) collectées en analysant des morceaux de terrains réels.



Analyse : les méthodes de synthèse par l'exemple sont relativement récentes. Beaucoup de ces techniques s'appuient sur des algorithmes de synthèse de textures par l'exemple. Ces méthodes ont pourtant du **mal à générer des structures hiérarchiques cohérentes** (*e. g.* les structures arborescentes de vallées). Les résultats générés dépendent fortement des bases de données de terrains réels fournies par l'utilisateur ce qui rend impossible la génération d'un relief qui n'est pas présent dans les exemples. Il est intéressant de noter que les données d'entrées correspondent souvent à des images d'une même résolution ; il serait intéressant d'avoir des mécanismes de génération par l'exemple qui décomposent le relief en plusieurs bandes de fréquences afin de pouvoir faire du transfert de style (les reliefs du Colorado avec le type de détails de l'Ohio).

2.3.4.2 Techniques d'esquisses et de dessins

Dans le cas d'une vue à la première personne au niveau du sol, la manière la plus simple de caractériser le relief est de décrire la silhouette de ce dernier. C'est pourquoi la modélisation par esquisses apparaît intuitive : l'utilisateur dessine la silhouette correspondant au relief souhaité et l'ordinateur propose un modèle 3D cohérent avec cette vision (Fig. 42).

Les travaux de WHELAN et VISVALINGAM [408] montrent que la notion de silhouettes d'un relief est plus complexe que prévu. Dans une vue ne comprenant que des esquisses, il est nécessaire d'améliorer la lisibilité de l'image en ajoutant des détails ou en prolongeant les silhouettes. Ils proposent un algorithme automatique d'extraction de silhouette qui ne repose pas seulement sur la notion d'occlusion mais prend en compte les pentes du terrain.

Les premières techniques d'esquisses de terrains remontent au logiciel HAROLD proposé en 2000, par COHEN *et al.* [75]. Si cette technique permet de créer une crête avec une altitude variable,

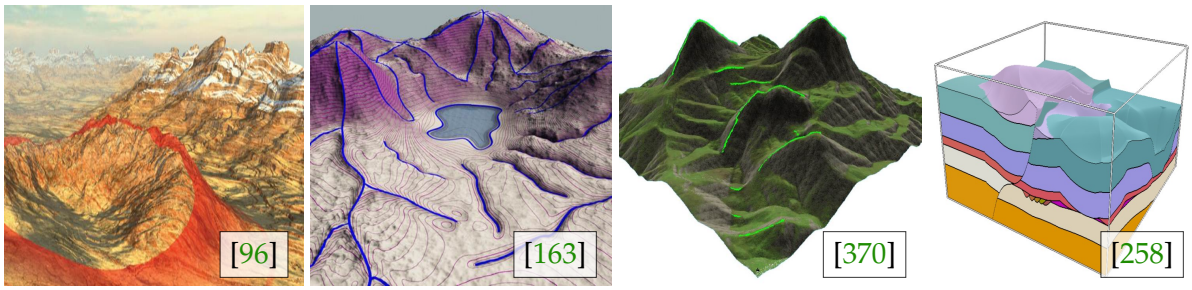


FIGURE 42 : Que ce soit avec des techniques d'esquisses ou des pinceaux intelligents, la modélisation de terrains surfaciques ou volumique peut devenir intuitive. Malgré tout, les terrains produits par les infographistes sont souvent planaires et ont une taille relativement petite.

la montagne qui la porte aura une profondeur fixe. WATANABE et IGARASHI [405, 406] étendent l'algorithme d'Harold en ajustant la profondeur et la forme du profil transversal en fonction de l'esquisse faite par l'utilisateur. VARLEY *et al.* [393] proposent un outil relativement simple et similaire.

Afin de rendre plus intuitifs et prédictibles les dessins de l'utilisateur, GAIN *et al.* [133] proposent de dessiner à la fois la silhouette des crêtes mais aussi l'empreinte au sol du massif montagneux. Le terrain est alors déformé automatiquement pour remplir cet espace. WÜNSCHE *et al.* [414] manipulent des lignes de contours esquissées par l'utilisateur ; ce dernier indique également le sens du gradient à l'aide de flèches. PUIG-CENTELLES *et al.* [309] utilisent deux vues en parallèles : l'une verticale pour dessiner les empreintes de morceaux de reliefs et pour indiquer des plans de coupes temporaires, et une autre qui permet de déformer le plan transversal actuel.

Les techniques d'esquisses peuvent être combinées avec de la synthèse de relief par l'exemple. En 2011, TASSE *et al.* [369, 372] donnent à l'utilisateur le moyen de contrôler la génération à l'aide d'esquisses. L'infographiste dessine un relief caractéristique (avec les outils proposés par GAIN *et al.* [133]) et ce morceau de terrain sera analysé et comparé à une base de données de paysages réels. Le reste du terrain sera généré procéduralement en utilisant [421] de façon à se combiner avec le morceau de relief dessiné. DOS PASSOS et IGARASHI [102] commencent par dessiner les silhouettes du relief, puis laissent leur algorithme trouver dans une base de données des morceaux de reliefs dont les crêtes correspondent à l'esquisse faite. Cette base de données est construite avec un ensemble de morceaux de terrains dont les crêtes sont extraites automatiquement selon plusieurs points de vue. En 2015, GAIN *et al.* [132] proposent un nouveau système pour mixer synthèse par l'exemple et définitions de contraintes par l'esquisse.

TASSE *et al.* [370, 371] continuent leur travaux en 2014 en proposant une méthode d'édition par esquisses à la première personne. Cette fois-ci, l'utilisateur dessine plusieurs silhouettes qui seront analysées, triées en profondeur à l'aide d'indices visuels, et mises en correspondance avec des crêtes du terrain actuel. Le relief est alors déformé de façon optimale pour correspondre aux nouvelles silhouettes de crêtes.

RUSNELL *et al.* [331] propose de représenter un terrain comme une collection d'esquisses au sol qui vont propager leur élévation. Chaque point du terrain se voit attribuer une altitude calculée

grâce à l'esquisse génératrice la plus proche (cette esquisse est trouvée grâce à l'algorithme de Djikstra qui calcule le chemin le plus cours dans la carte de hauteurs).

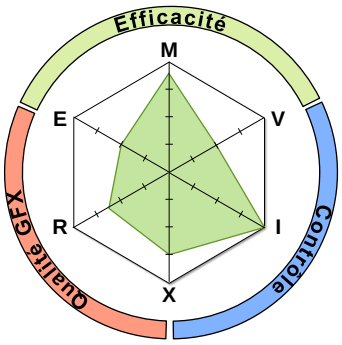
HNAIDI *et al.* [162, 163] s'inspirent des courbes de diffusion (*diffusion curves*) présentées par ORZAN *et al.* [274] pour le dessin vectoriel. Ils adaptent cette approche à la modélisation de terrains en laissant l'utilisateur dessiner un ensemble de courbes (et des profils d'altitude qui leur sont associés). Un algorithme de diffusion multi-résolutions implémenté sur GPU construit alors la carte de hauteurs associée à ces informations. BERNHARDT *et al.* [35, 36] complètent ces travaux avec une interface facilitant le contrôle de la méthode ; ils utilisent aussi une structure adaptative en *quadtree* gérée par le CPU pour accélérer les calculs. TAKAYAMA *et al.* [367] proposent les surfaces de diffusion (*diffusion surfaces*) qui sont une extension des courbes de diffusion pour construire des objets volumiques 3D ; ils donnent plusieurs exemples de résultats dont une scène naturelle représentant un volcan.

Plusieurs techniques d'esquisses ont également été proposées pour modéliser la structure géologique interne. En 2012, NATALI *et al.* [259] permettent à l'utilisateur de dessiner sur un plan des couches géologiques superposées et proposent des outils spécifiques pour modéliser les failles et les plis ainsi que pour déformer chaque couche du terrain. Dans un premier temps, ces travaux seront repris par BENDIKSEN [26, 27] qui adaptera ces techniques d'esquisses pour construire des morceaux 3D de terrains. NATALI *et al.* [258] étendent à nouveau ce modèle pour calculer sur GPU et visualiser sous forme d'animations les mouvements géologiques.

Au lieu de dessiner des silhouettes de crêtes, DE CARPENTIER et BIDARRA [95, 96] adaptent les pinceaux virtuels que l'on peut retrouver dans des éditeurs d'images (THE GIMP, PHOTOSHOP) pour pouvoir modéliser facilement des terrains. Les brosses qu'ils proposent sont intelligentes : elles ne changent pas simplement l'élévation (comme quand on manipule une carte de hauteurs) mais indiquent quel type de relief l'utilisateur souhaite (il modifie les paramètres locaux de génération du terrain).

Un *framework* mixte d'édition de terrain a été proposé par BRADBURY *et al.* [55]. Il combine plusieurs approches dont des techniques d'esquisses, des outils de dessins avec pinceaux mais aussi des techniques de décomposition du terrain selon plusieurs bandes de fréquences pour pouvoir transférer certaines caractéristiques depuis un terrain de référence (les détails d'un terrain réel, ou la structure d'une carte de hauteurs).

Il est possible de visualiser les terrains de façon non réaliste, en particulier dans le cas de la visualisation de panoramas (virtuels ou réels). JENNY *et al.* [177] arrivent à améliorer la lisibilité d'une prise de vue d'un terrain en utilisant certains types de projection de type perspective. Ils modifient également le relief en exagérant l'altitude (en fonction de la distance à la caméra) et en appliquant des déformation locales pour agrandir des éléments distinctifs comme des vallées.



Analyse : les méthodes d'esquisses se sont développées aux cours des dernières années. Elles ont été rapidement adoptées dans des logiciels de sculpture car **très intuitives pour les utilisateurs** (qui ont souvent suivi une formation d'infographie 2D). Ces méthodes gagneraient à être plus rapides (pour avoir un contrôle interactif des déformations) et à gérer plusieurs échelles en même temps (pour, par exemple, esquisser quelques détails et laisser un algorithme remplir le reste de la scène avec le même genre de micro-reliefs). Il serait intéressant d'adapter ces méthodes d'esquisses pour pouvoir construire facilement la trajectoire d'une rivière sur le relief, décorer à l'aide d'instances le paysage ou même définir des textures.

2.3.4.3 Contrôle vectoriel ou sémantique

Il est possible de construire un paysage en utilisant des outils de plus haut niveau. Utiliser des données vectorielles permet de manipuler des objets sans définir la géométrie finale. Dans le cadre des systèmes d'informations géographiques, on manipule souvent plusieurs calques d'informations (le relief, la végétation, les trajectoires de rivières, de routes, ...). Avec des données vectorielles, on peut également agir au niveau sémantique pour rendre cohérentes les scènes créées (Fig. 43).

Bruneton et Neyret [60] enrichissent des terrains avec des données vectorielles représentant des rivières, des routes et des champs. Ils visualisent ensuite les terrains à l'aide de *quadtree* adaptatifs. Ces données sont éditables en temps-réel et permettent d'obtenir des maillages fins sans artefacts graphiques (par exemple, pour visualiser les bords des rivières correctement, ce qui est impossible avec des données satellites).

Des approches basées sur la modélisation déclarative existent : Smelik *et al.* [44, 349, 350, 351, 352, 353] permettent à l'artiste de dessiner avec une image bitmap ou de manière vectorielle des informations sémantiques (l'empreinte au sol d'une ville, la trajectoire d'une rivière) et utilisent une collection de techniques procédurales pour créer géométriquement ces éléments. Ils proposent également un algorithme pour vérifier la cohérence (*consistency*) du monde virtuel et des règles de résolution de conflits (en fonction d'une priorité). Ces travaux, combinés à de nombreuses méthodes procédurales, sont détaillés dans la thèse de Smelik [354].

Une approche similaire est tentée par Hultquist *et al.* [168] qui cherchent à créer des scènes naturelles en laissant l'utilisateur décrire l'atmosphère, et en pondérant avec plusieurs qualificatifs (calme, doux, orageux, humide, ...) qui vont influencer à la fois la génération du relief, la configuration de l'éclairage, et le type de textures utilisées.

Des représentations abstraites peuvent aussi être utilisées pour représenter des éléments très spécifiques comme les cascades. Emilien *et al.* [111, 112] manipulent un graphe représentant les différents écoulements d'eau que l'utilisateur souhaite obtenir. Après une analyse des flux et une catégorisation des écoulements, l'algorithme construit automatiquement les rivières et cascades correspondantes à l'aide de modèles géométriques paramétrés.

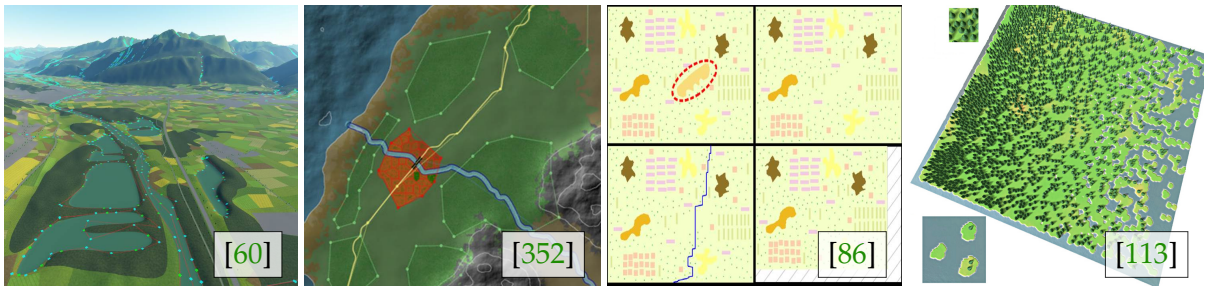
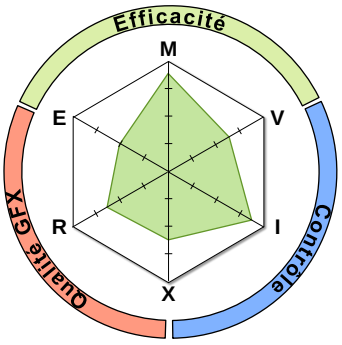


FIGURE 43 : Les méthodes s'appuyant sur des données vectorielles ou sémantiques permettent à l'infographiste de manipuler des objets de plus haut niveau et de s'affranchir du calcul de la géométrie.

Le redimensionnement de scènes intelligentes est également une voie pour la synthèse/réorganisation d'environnements. Le recadrage intelligent (*seam carving*) [14] est une technique basée image qui cherche à redimensionner de manière optimale une texture en lui enlevant les bandes de pixels les moins importantes d'un point de vue visuel. CRUZ *et al.* [86] proposent en 2014 une extension pour agrandir ou réduire des scènes définies de manière vectorielle.

En 2015, EMILIEN *et al.* [113] parviennent à réaliser un outil intelligent de peinture basé sur une modélisation procédurale inverse. En analysant localement un morceau d'une scène complexe crée avec des outils classiques, ils retrouvent les paramètres statistiques qui décrivent les interactions entre les différents types d'objets. Leur approche mêle techniques de peintures, modélisation inverse, génération procédurale et édition de haut-niveau.

Analyse : les techniques à base de données vectorielles ou de modélisation déclarative permettent à des néophytes de générer des scènes virtuelles de manière automatique. Ces utilisateurs n'ont alors **pas besoin de s'intéresser aux notions de géométrie** ou de *texturing*. Ces méthodes fournissent généralement un *framework* unifié qui permet d'utiliser plusieurs techniques (génération procédurale, esquisse, description sémantique) pour créer un paysage. Fonctionnant avec des données vectorielles, ces algorithmes sont capables d'éditer de grands terrains avec différentes couches d'informations (végétation, trajectoires des rivières, peuplement, *etc*).



2.3.4.4 Autres techniques de contrôle

D'autres techniques de contrôles existent pour modifier un terrain (Fig. 44). En 2005, STACHNIAK et STUERZLINGER [357] proposent de modifier n'importe quel terrain avec un ensemble de contraintes (*e. g.* une courbe 3D par laquelle le terrain doit passer). Leur algorithme va chercher à optimiser le terrain d'entrée pour satisfaire cette contrainte et s'autorisera à contraindre itérativement des points et à pratiquer de petites déformations locales dans leur voisinage.

DACHSBACHER et STAMMINGER travaillent sur une approche [90] qui cherche à améliorer la visualisation d'un terrain en ajoutant des détails géométriques procéduraux au moment du rendu.

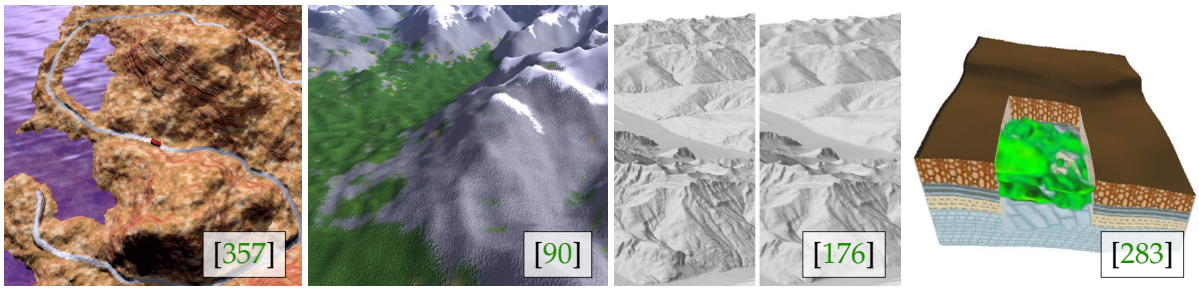


FIGURE 44 : Quelques techniques de contrôles sont difficiles à classer parce qu'elles agissent sur des éléments précis (le niveau de détails, la déformation du terrain, l'interprétation de données géologiques).

Ces détails sont ajoutés et placés en fonction de la distance de la caméra au sol, de l'orientation du relief, *etc*.

JENNY *et al.* [176] proposent un équivalent aux *equalizers* qu'on utilise dans les logiciels de traitement du son. Leur logiciel permet d'accentuer ou de diminuer certaines fréquences du terrain (les reliefs importants ou les détails géométriques) en manipulant une pyramide de cartes de hauteurs (correspondant aux différents niveaux de *mipmapping* du terrain) et une pyramide des contours (correspondant aux Laplaciens du terrain).

De nombreuses techniques de contrôle propres aux systèmes de modélisation de géologie existent. CAUMON *et al.* [66] analysent le *workflow* nécessaire pour convertir des données géologiques en un maillage visualisable. D'après LIDAL [220], créer un outil unique de modélisation géologique est un problème très complexe – plus ardu que dans d'autres domaines comme l'architecture – car il y a besoin d'une notion de temporalité, et parce qu'il existe plusieurs façon de représenter un même objet géologique. BÉZIN *et al.* [41] tentent de créer un tel outil et proposent un *framework* complet pour modéliser des objets géologiques de façon intuitive. Certains s'intéressent à modéliser les mouvements géologiques sous la forme d'un historique d'événements [56, 221] ; d'autres systèmes aident à texturer les différentes couches géologiques [283, 284] ; et certains cherchent à faciliter la création d'animations [212].

2.4 Conclusions

Si la modélisation de terrains est un « vieux » sujet d'étude, les axes de recherches ont beaucoup évolué au cours du temps. Par manque de puissance des ordinateurs pour simuler les phénomènes physiques, les premières techniques proposées étaient principalement **des méthodes procédurales**. Avec l'augmentation de la puissance des CPU et la volonté de faire des images toujours plus réalistes (*i. e.* un objectif de qualité), beaucoup de travaux se sont intéressés à proposer **des simulations physiques** de différents phénomènes (érosion hydrique, changements de température, *etc*). Ces travaux ont été confortés par l'avènement des cartes graphiques qui ont accéléré les calculs de ces simulations. Plus récemment, c'est sur **le contrôle** que se sont penchés les chercheurs et les industriels. Le besoin de créer plus d'environnements, plus grands, tout en

maintenant une qualité graphique homogène (*i. e.* un objectif de quantité), poussent à développer des méthodes de plus haut niveau, plus intuitives (comme les techniques d'esquisses).

L'axe de recherche que nous poursuivons dans cette thèse revient à donner des **moyens de contrôler les méthodes procédurales** à la fois en utilisant **des représentations adaptées** mais aussi en proposant **de nouveaux algorithmes de génération** qui cherchent à imiter la nature sans la simuler.

D'autres **manques peuvent se dégager** de cet état de l'art. S'il est évidemment possible de s'intéresser à l'optimisation de telle ou telle méthode, **certains phénomènes spécifiques n'ont que peu été étudiés** (l'utilisation de la tectonique ou la modélisation des glaciers) voire pas du tout (l'érosion des roches par la foudre ou par le vent).

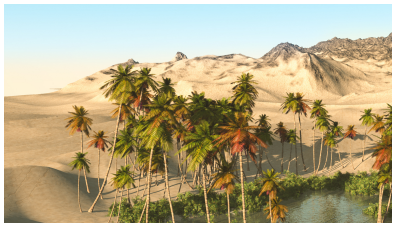
Pour l'instant, deux voies d'études restent largement ouvertes. D'une part, **l'interopérabilité des méthodes et des structures de données** reste encore à approfondir. Les simulations (d'érosions hydriques ou éoliennes) travaillent presque uniquement avec des cartes de hauteurs et ne sont presque jamais implémentées au sein d'un *framework* unifié. Ces simulations manipulent généralement des objets (l'eau, le vent) qui ont des propriétés (la température, la vitesse) pouvant influencer de nombreux autres éléments de la génération (la végétation, l'apparence).

L'autre voie concerne la notion d'animation ou de **modifications liées au temps**. Ces modifications peuvent se produire à différentes échelles : ainsi un débordement de rivière résultant en une inondation dure quelques jours alors que l'érosion glaciaire va agir sur des milliers d'années. Gérer la temporalité et pouvoir extraire des modèles qui sont paramétrables (« je veux obtenir la forme qu'aura cette rivière dans 50 ans ») est un problème encore relativement peu étudié.

Enfin, la modélisation volumique des terrains n'intéresse pour l'instant que les spécialistes des géo-sciences mais il est évident que **les structures volumiques devraient progressivement prendre le pas** sur les modèles surfaciques pour des raisons de réalisme (les simulations pourront modéliser des écoulements internes, des changements d'état, *etc*) ou pour des raisons de *gameplay* (les joueurs ou les artistes pourront modifier interactivement la structure du monde). Pour l'instant, **il n'existe pas encore de représentation volumique efficace, manipulable et légère en mémoire**. Si on parvient à optimiser davantage les SVO pour l'animation et la simulation, il est possible que dans un futur proche, les voxels – par leur simplicité d'utilisation – remplacent les modèles triangulaires planaires ou non.

3

Modélisation par modelés



Sommaire

Simple things should be simple,
complex things should be possible.

— Alan Key

Dans ce chapitre, nous présentons un modèle de représentation de terrains, s'inspirant des surfaces implicites et de la CSG (*Constructive Solid Geometry*). Cette représentation est adaptée à l'édition de terrains grâce à ses primitives à supports compacts qui permettent de sculpter le terrain localement. Elle permet de décrire un relief complexe en manipulant des primitives qui modélisent des parcelles de terrains dont le relief est homogène (*landform features*) [1].

Après avoir présenté le modèle de représentation de terrains planaires et continus par arbres de construction (Section 3.1), nous détaillons les primitives (Section 3.2) et opérateurs (Section 3.3) utilisables pour construire des scènes naturelles. Nous étendons ensuite notre modèle en un graphe acyclique orienté pour gérer l'instanciation de morceaux de terrains (Section 3.4) à l'aide d'opérateurs de transformations spécifiques. Le système comporte un mécanisme de niveaux de détails (Section 3.5) grâce à des primitives ou des opérateurs spécifiques. Nous montrons également comment étendre le modèle pour représenter des terrains non-planaires et/ou multi-couches (Section 3.6), avant de présenter quelques exemples de scènes construites avec ce modèle (Section 3.7). Nous abordons ensuite les limitations et les perspectives de cette représentation (Section 3.8). Enfin, nous concluons ce chapitre (Section 3.9) avec un résumé des principaux avantages et inconvénients du modèle de terrains.

Ces travaux ont fait l'objet de deux publications [141, 142].

1. On parle en géographie/géomorphologie de « modelés ».

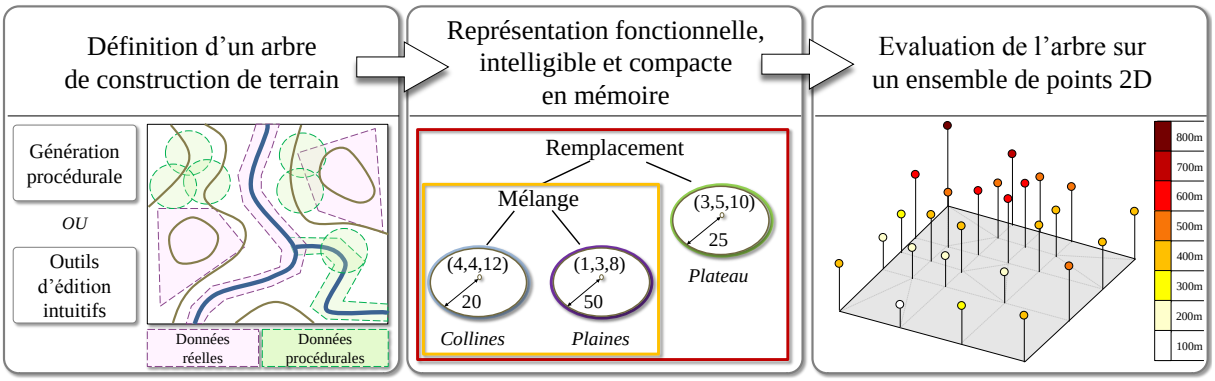


FIGURE 45 : Ce modèle de représentation permet de combiner à la fois des données réelles et des techniques procédurales. La structure repose sur un arbre de construction où chaque feuille correspond à une primitive (qui décrit une *landform feature*). Une fois l'arbre construit, il est possible de l'échantillonner afin de visualiser le terrain décrit.

3.1 Modèle vectoriel de terrains

Nous proposons de représenter un terrain par un arbre de construction composé de primitives et d'opérateurs qui sera évalué pour obtenir un relief (Fig. 45). Cette structure est vectorielle : elle ne stocke pas un ensemble d'échantillons de la surface (comme dans le cas d'une carte de hauteurs ou d'un TIN) mais utilise des primitives qui décrivent localement le relief. Chaque arbre représente donc un terrain défini sur un domaine et dont le relief est continu. Les primitives étant définies avec des supports compacts, chaque sous-arbre peut donc être vu comme un morceau de terrain (qui peut être combiné avec d'autres morceaux pour obtenir le relief final).

3.1.1 Structure générale

Le modèle de représentation est basé sur un arbre composé de primitives et d'opérateurs (Fig. 46). Chaque arbre ou sous-arbre correspond à un terrain défini sur un domaine spatial donné. Un arbre de construction permet de facilement représenter et construire l'expression algébrique correspondant à la surface du relief. C'est, intrinsèquement, une représentation procédurale.

L'arbre est composé de deux types de nœuds représentant des éléments de nature différente : d'une part les feuilles et d'autre part les nœuds internes.

Les feuilles de l'arbre agissent comme des primitives générant chacune un morceau de terrain. Généralement cette portion de terrain correspond une unité de relief homogène qui peut contenir des éléments géologiques ou géographiques caractéristiques (*landform features*).

Les nœuds internes de l'arbre représentent des opérateurs qui vont combiner et agréger leurs sous-arbres (chaque sous-arbre représentant récursivement un morceau de terrain composé *in fine* de une ou plusieurs primitives).

Étant donné un arbre, l'élévation d'un point $\mathbf{p}$ est calculée en évaluant la combinaison hiérarchique de toutes les primitives de l'arbre : ce processus est nommé évaluation de l'arbre.

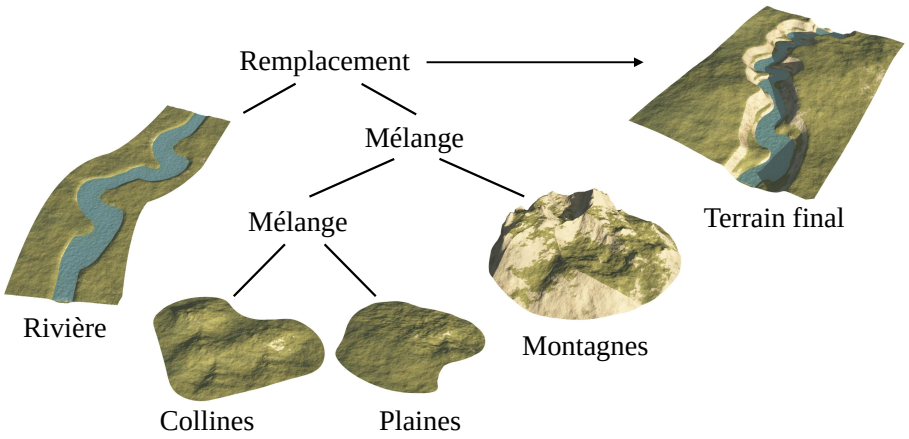


FIGURE 46 : Un arbre de construction simplifié représentant une rivière creusée dans un relief accidenté. Éditer la scène revient à changer la structure de l'arbre ou à modifier le placement des primitives et à manipuler les paramètres des différentes primitives (bruits, domaines, ...).

3.1.2 Structure des nœuds

Nous définissons deux fonctions f et α pour chaque nœud. L'évaluation de l'arbre calcule et combine récursivement les contributions de chaque nœud. En pratique, on manipule toujours simultanément des paires de valeurs $(f(\mathbf{p}), \alpha(\mathbf{p}))$.

La fonction d'élévation $f : \mathbf{R}^2 \to \mathbf{R}$ définit l'élévation du terrain. Dans le cas d'une primitive, ce calcul peut s'appuyer sur des paramètres contenus dans la structure du nœud comme le type de la fonction de bruit. Dans le cas d'un opérateur, la fonction d'élévation du nœud est calculée à l'aide d'une combinaison des fonctions d'élévation et des fonctions de poids que l'on retrouve dans les nœuds enfants.

La fonction de poids $\alpha : \mathbf{R}^2 \to \mathbf{R}^+$ définit comment le terrain sera combiné à son environnement (c'est-à-dire avec d'autres morceaux) à l'aide d'un opérateur.

3.1.3 Domaines de définition et continuité

Nous définissons Ω_0 comme le support de la fonction de poids α :

$$\Omega_0 = \{\, \mathbf{p} \in \mathbf{R}^2 \mid \alpha(\mathbf{p}) > 0 \,\}.$$

Le support d'un arbre de construction est compact à partir du moment où toutes les fonctions α combinées le sont aussi.

La fonction d'élévation f est définie sur le domaine Ω_0. Soit $T > 0$ une valeur de seuil : le terrain est défini sur le sous-domaine noté $\Omega_T \subset \Omega_0$:

$$\Omega_T = \{\, \mathbf{p} \in \mathbf{R}^2 \mid \alpha(\mathbf{p}) \geq T > 0 \,\}.$$

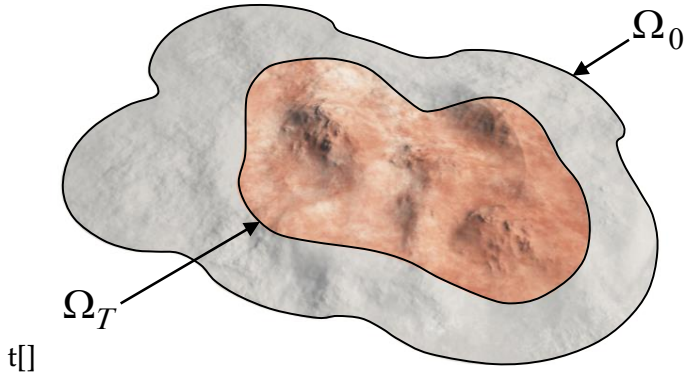


FIGURE 47 : Caractérisation du domaine : Ω_0 correspond au domaine de définition du morceau de terrain (c'est-à-dire le support compact de la fonction α). Le terrain que l'on extrait existe dans le domaine Ω_T avec $\Omega_T \subset \Omega_0$. La zone intermédiaire $\Omega_0 \setminus \Omega_T$ correspond au domaine d'influence qui permet de garantir aux fonctions f de bonnes propriétés de continuité.

Un exemple présenté dans la Fig. 47 illustre ce concept et représente à la fois Ω_0 et $\Omega_T \subset \Omega_0$. À noter que les deux domaines Ω_0 et Ω_T peuvent être composés d'une ou plusieurs composantes connexes. Dans le reste des illustrations, la région Ω_0 ne sera pas montrée sauf mention contraire explicite.

Dans notre implémentation, nous utilisons $T = 1/2$. Nous définissons la surface du terrain T comme l'image de Ω_T par la fonction f :

$$\mathrm{T} = \{\, (\mathbf{p}, f(\mathbf{p})) \in \mathbf{R}^3, \mathbf{p} \in \Omega_T \,\}.$$

Ce modèle de représentation par arbres permet d'adapter le calcul de la fonction d'élévation f pour obtenir un mécanisme de niveau de détails continu. De plus, en construisant progressivement la structure hiérarchique, nous pouvons garantir certaines propriétés mathématiques : la fonction d'élévation f sera Lipschitzienne pour chaque primitive ou pour chaque opérateur. La fonction de poids α sera C^1. Ces propriétés différentielles nous permettent d'accélérer les calculs de visualisation des terrains (cf. Section 4).

3.1.4 Structures hiérarchiques

L'arbre de construction intègre nativement un système hiérarchique de boîtes englobantes (BVH ou *Bounding Volume Hierarchy*) qui permet facilement de rechercher les nœuds qui contribuent au calcul d'élévation d'un point $\mathbf{p}$ du terrain. Grâce à cette propriété, nous évaluons les terrains de manière efficace en élaguant l'arbre dynamiquement et spatialement. En pratique, l'évaluation d'un point du terrain ne fait intervenir que peu de primitives (seulement celles dont le support compact recouvre ce point).

Cette propriété permet également de décomposer un arbre de construction en un ensemble de plusieurs arbres de construction simplifiés selon leur position dans l'espace. Cela correspond à un élagage spatial pré-calculé statiquement. Le découpage se fait généralement selon

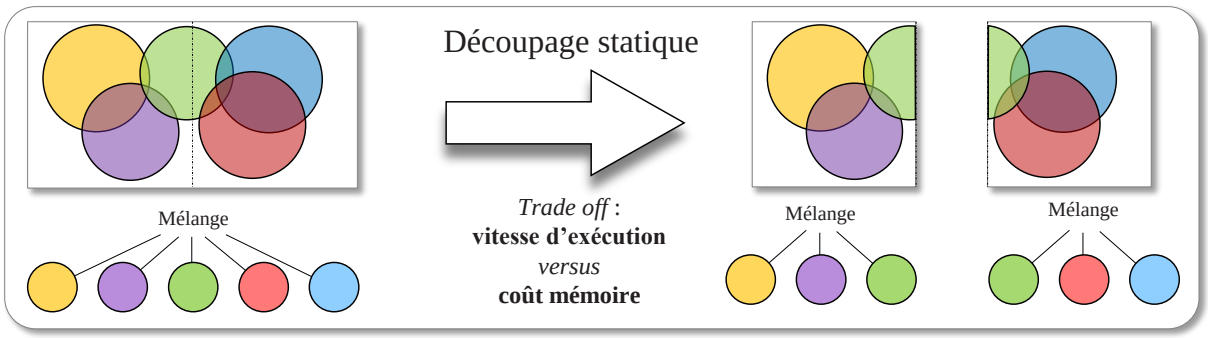


FIGURE 48 : Si un arbre de construction correspond à l'ensemble du terrain, il est possible de le découper statiquement en plusieurs arbres qui définissent chacun une parcelle du domaine (par exemple, selon une grille). Les nouveaux arbres sont simplifiés en supprimant les primitives qui ne contribuent pas au calcul du relief de cette parcelle. Même si certaines primitives sont dupliquées dans plusieurs arbres (entraînant un coût-mémoire), les performances sont grandement améliorées car l'élagage spatial ne se fera pas dynamiquement à l'exécution (au travers de tests de boîtes englobantes) mais statiquement au moment de la découpe.

une grille 2D. Pour chaque case, on calcule le sous-arbre (Fig. 48) correspondant à l'ensemble des primitives qui intersectent la case de la grille. Ce sous-arbre décrit le morceau de terrain correspondant à la case de la grille avec une continuité au niveau des bords. Chaque sous-arbre correspond à une simplification : les termes/primitives n'apparaissant jamais dans le calcul du morceau de domaine sont éliminés. Ce processus permet de diminuer les temps de calculs pour un coût mémoire relativement faible (la structure sous forme d'arbre étant peu gourmande en mémoire).

3.1.5 Évaluation des normales

Afin de visualiser les scènes avec un éclairage réaliste, il est nécessaire de calculer la normale au terrain en un point $\mathbf{p}$ donné. Celle-ci peut être calculée soit par l'évaluation du gradient analytique de l'élévation $\nabla f(\mathbf{p})$, soit par une approximation discrète usuellement réalisée par un calcul par différences finies.

$$\nabla f(\mathbf{p}) \simeq \frac{1}{2\varepsilon} \begin{pmatrix} f(\mathbf{p}+\varepsilon_x) - f(\mathbf{p}-\varepsilon_x) \\ f(\mathbf{p}+\varepsilon_y) - f(\mathbf{p}-\varepsilon_y) \end{pmatrix}.$$

Notre système implémente un calcul analytique du gradient pour certains nœuds (ceux dont on a trouvé une définition analytique de la normale). Lorsque l'évaluation de $f(\mathbf{p})$, $\alpha(\mathbf{p})$ et du gradient $\nabla f(\mathbf{p})$ sont nécessaires, nous utilisons une requête optimisée qui, lors d'une évaluation, calcule et renvoie simultanément l'élévation et la normale en un seul appel. Cette optimisation améliore les temps de calculs, non seulement parce que l'arbre est parcouru une seule fois, mais également parce que de nombreux termes communs sont factorisés.

3.2 Primitives

Nous proposons principalement deux types de primitives : des primitives procédurales et des primitives construites à partir d'images. Ces deux types permettent un contrôle complémentaire quant à la gestion du placement des éléments caractéristiques. Les primitives procédurales ont l'avantage d'être légères en consommation mémoire et rapides à évaluer mais elles nécessitent de trouver une fonction mathématique qui imite bien le relief du modelé qu'on souhaite représenter. À l'inverse, les primitives à images permettent, en contrepartie d'un coût- mémoire beaucoup plus important, d'intégrer à l'intérieur d'un terrain, des données réelles ou des reliefs créés par un artiste de façon plus traditionnelle.

Ces deux modèles de primitives s'appuient sur des squelettes pour définir des domaines compacts ou être à support infini. Les primitives à squelettes sont inspirées à la fois de la CSG (Constructive Solid Geometry) et du modèle du Blob-Tree [415]. Ces primitives sont définies par un squelette géométrique (*e. g.* un point, un segment, une courbe ou encore un contour) et par un ensemble de paramètres qui décrivent les fonctions d'élévation et de poids en fonction de la distance au squelette. Nous utilisons différents types de squelettes adaptés à la représentation des éléments caractéristiques d'un terrain : disques, courbes, quadrangles, contours polygonaux.

L'utilisation de primitives à supports infinis (c'est-à-dire dont la fonction de poids α n'est pas à support compact) est tout à fait possible dans ce modèle. On peut par exemple utiliser des primitives dont le domaine est défini à l'aide d'une fonction de bruit. Toutefois, comme leur influence n'est pas limitée spatialement, il sera nécessaire de calculer leur contribution (qui peut être nulle) pour chaque évaluation (ce qui a un impact sur les performances). De la même manière, ces primitives agissant sur l'ensemble du relief, leur utilisation impactera le calcul de certaines propriétés mathématiques utilisées pour la visualisation (sur-évaluant la borne de Lipschitz).

Pour des raisons d'efficacité, les fonctions d'élévation et de poids ne doivent pas nécessiter des procédures de calculs longues et complexes. Ce modèle de terrains hiérarchique correspond à une définition fonctionnelle et *pure* de l'élévation qui permet de renvoyer une valeur en tout point du domaine en un temps relativement court. De même, l'utilisation de primitives à support infini ne permettra pas d'élaguer spatialement l'arbre de construction : nous cherchons donc plutôt à ce que tout point de l'espace ne soit influencé que par un nombre restreint de primitives [2].

Fonctions d'élévations

Pour chaque primitive, nous utilisons une fonction η qui imite la fonction de relief d'un modelé. Dans la plupart des primitives, η est calculé à l'aide de fonctions de bruits multi-harmoniques n_H (qui représentent bien les zones montagneuses). Dans notre implémentation, nous utilisons η_m (qui correspond à une fonction *ridged-multifractal*) pour définir des montagnes, η_c (fonction *turbulence*) pour définir des collines et η_p (fonction constante) pour définir des plaines.

2. En pratique, dans les différentes scènes de ce chapitre, un point du domaine est recouvert par une dizaine de primitives au maximum

Soit $\mathrm{n}(\mathbf{p}) : \mathbf{R}^2 \to \mathbf{R}$ la fonction de bruit à une unique harmonique, h le nombre d'harmoniques souhaité, $\{a_i\}$ un ensemble d'amplitudes de valeurs décroissantes, et $\{s_i\}$ un ensemble de fréquences croissantes.

$$\eta_m(\mathbf{p}) = \sum_{i=1}^{h} a_i \ \mathrm{n}(\mathbf{p} \times s_i) \ , \qquad \eta_{m'}(\mathbf{p}) = \sum_{i=1}^{h} a_i \left(1 - |\mathrm{n}(\mathbf{p} \times s_i)|\right) ,$$
$$\eta_c(\mathbf{p}) = \sum_{i=1}^{h} a_i \, |\mathrm{n}(\mathbf{p} \times s_i)| \ , \qquad \eta_p(\mathbf{p}) = 0 \ .$$

De Carpentier et Bidarra [96] proposent de généraliser la formule de somme de bruits pour y intégrer des transformations non linéaires qui permettent de créer des formes plus complexes. On a alors la formule :

$$\eta(\mathbf{p}) = T_{post}\left(\sum_{i=1}^{h} a_i \, T_{in}\Big(\mathrm{n}\big(T_{pre}(\mathbf{p} \times s_i)\big)\Big)\right)$$

où T_{pre}, T_{in} et T_{post} sont des fonctions de transformation paramétrables par l'utilisateur. On rappelle que $T_{pre} : \mathbf{R}^2 \to \mathbf{R}^2$, $T_{in} : \mathbf{R} \to \mathbf{R}$ et $T_{post} : \mathbf{R} \to \mathbf{R}$. Généralement T_{pre} correspond à une fonction de déformation (*warping*) qui utilise une ou plusieurs fonctions de bruit.

Fonctions de poids

Les fonctions de poids α sont définies comme des fonctions continues C^1 décroissantes sur un support compact pour limiter l'influence de la primitive et pour contrôler la manière dont elles seront combinées par les opérateurs de l'arbre de construction. Soit $d(\mathbf{p})$ la distance entre le point évalué $\mathbf{p}$ et le squelette. Nous définissons la fonction de poids α comme la composition de la distance avec un filtre lisse décroissant g :

$$\alpha(\mathbf{p}) = g \circ d(\mathbf{p}) .$$

Dans notre implémentation, nous utilisons la fonction g_n définie par Wyvill *et al.* [415] qui permet un contrôle flexible. Cette fonction permet d'obtenir une classe de continuité C^n avec $n \geq 0$ selon le choix de l'exposant :

$$g_n(x) = \begin{cases} \left(1 - \left(\frac{x}{r}\right)^2\right)^{n+1} & \text{si } x < r, \\ 0 & \text{sinon.} \end{cases}$$

Nous utilisons généralement les fonctions g_2 et g_3 qui produisent des mélanges lisses.

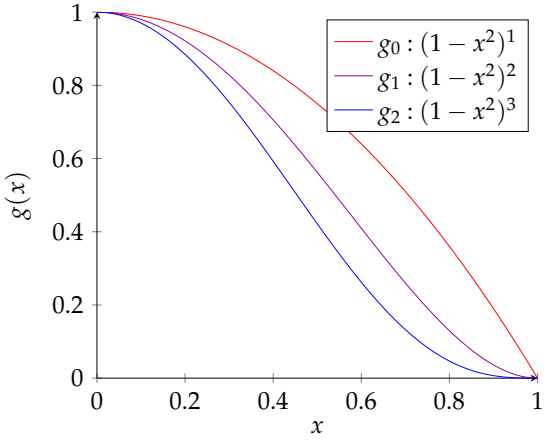


Figure 49 : Les fonctions de mélange de Wyvill.

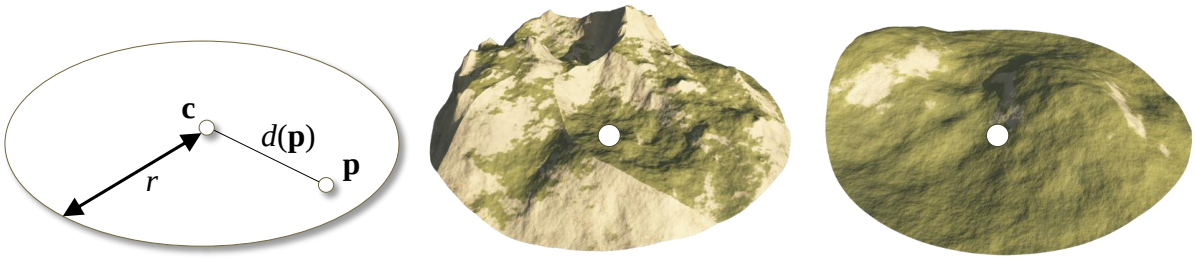


FIGURE 50 : Les squelettes ponctuels ou à disques permettent de générer de petites zones circulaires qui se combinent pour former un terrain plus grand. Plusieurs fonctions d'élévation peuvent s'attacher au domaine : une fonction constante donnera un morceau de plaine, une turbulence produira une colline et une fonction de *ridge-multifractals* imitera le relief d'une montagne.

Fonctions images

Les fonctions construites à partir de données discrètes (comme des images) sont utilisées pour définir des éléments caractéristiques des terrains spécifiques qui sont difficiles à modéliser par une fonction analytique, comme des morceaux de rivières détaillés ou des ridules de sables.

Les fonctions d'élévation et de poids sont définies en projetant des cartes de hauteurs et de poids sur un domaine Ω dont on connaît une paramétrisation (u,v). L'évaluation de f et α se fait alors en interpolant les données pour garantir que f soit Lipschitzienne et que α soit au moins de classe C^1. Dans notre implémentation, nous utilisons des interpolations bilinéaires ou bicubiques contraintes.

3.2.1 Les primitives ponctuelles

Ces primitives représentent des petites portions de terrain. Les squelettes ponctuels ont un centre $\mathbf{c} = (\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_z)$ et un rayon r qui décrivent la zone d'influence de la primitive (Fig. 50). Nous notons la fonction de relief $\eta(\mathbf{p})$: elle est paramétrée pour contrôler la rugosité locale du terrain. Nous définissons :

$$f(\mathbf{p}) = \mathbf{c}_z + \eta(\mathbf{p} - \mathbf{c}_{xy}). \qquad \alpha(\mathbf{p}) = \begin{cases} \left(1 - \left(\frac{d(\mathbf{p})}{r}\right)^2\right)^3 & \text{si } d(\mathbf{p}) < r, \\ 0 & \text{sinon.} \end{cases}$$

L'utilisation d'un point 3D permet de définir une petite surface de terrain plate à une altitude donnée (calcul que l'on retrouve dans l'utilisation de $\mathbf{c}_z$). Il est également envisageable de donner une influence anisotrope à ces primitives, à travers la définition de squelettes ellipsoïdaux.

3.2.2 Les primitives à disque

Ces primitives représentent également des petites portions de terrain mais comportent une zone où leur influence est constante. Leurs squelettes sont décrits par un centre $\mathbf{c} = (\mathbf{c}_x, \mathbf{c}_y, \mathbf{c}_z)$ et un rayon r qui décrivent la zone d'influence de la primitive ainsi que par un rayon intérieur r_{int}.

$$f(\mathbf{p}) = \mathbf{c}_z + \eta(\mathbf{p} - \mathbf{c}_{xy}). \qquad \alpha(\mathbf{p}) = \begin{cases} 1 & \text{si } d(\mathbf{p}) < r_{int}, \\ \left(1 - \left(\dfrac{d(\mathbf{p}) - r_{int}}{r - r_{int}}\right)^2\right)^3 & \text{si } r_{int} \leq d(\mathbf{p}) < r, \\ 0 & \text{sinon}. \end{cases}$$

Il est possible de modifier la primitive afin que la valeur de poids dans la zone centrale corresponde à une valeur précise (différente de 1) en multipliant le résultat de la fonction par cette valeur.

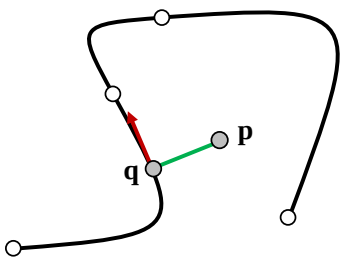
Ces primitives à disques sont une généralisation des primitives-ponctuelles très utiles lorsqu'on les combine avec un opérateur de remplacement (cf. Section 3.3). En effet, la primitive-circulaire va généralement avoir un domaine où sa fonction de poids vaudra 1 qui n'est pas forcément réduit à un point. Dans le cas d'un opérateur de remplacement ou de dépôt, l'effet de modification du terrain dans cette zone sera intuitif (il ne correspond pas à une zone de transition où les résultats sont moins prévisibles).

3.2.3 Les primitives courbes

Ces primitives sont construites à l'aide d'une courbe notée Γ définie par morceaux, et d'un ensemble de profils $\{c_i\}$ qui décrivent le relief sous forme de coupes transversales (Fig. 51). Nous stockons chaque plan de coupe comme une fonction mono-dimensionnelle par morceaux afin d'avoir un calcul rapide d'une élévation unique. Ces plans de coupe peuvent être utilisés aussi bien pour décrire les profils de la fonction d'élévation que ceux de la fonction de poids $\{c\_\alpha_i\}$.

L'utilisation de lignes polygonales (c'est-à-dire de degré 1) permet de réaliser les calculs très rapidement mais seule la continuité C^0 est garantie. Il est donc nécessaire d'utiliser des courbes de plus haut degré qui permettent de représenter des trajectoires complexes.

Généralement, nous décomposons la courbe en un ensemble de morceaux de quadriques (c'est-à-dire de degré 2). La distance d'un point à la courbe complète correspond au minimum des distances à chaque morceau. De plus, le calcul de la projection $\mathbf{q}$ d'un point $\mathbf{p}$ sur une courbe quadratique Γ_i définie entre les points Γ_i^0 et Γ_i^1 possède une formulation exacte [233] : cela revient à calculer la racine d'une équation cubique [232]. Nous avons besoin de



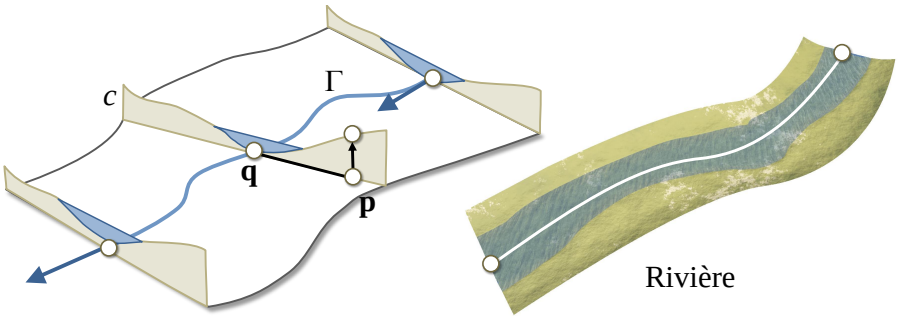


FIGURE 51 : Les primitives-courbes permettent de facilement créer des reliefs suivant une trajectoire comme des routes ou des rivières. Les courbes peuvent être construites par morceaux de segments ou de splines mais, pour une utilisation pratique, il est important d'avoir une fonction de distance à ce squelette qui soit rapide à évaluer. Si on souhaiter attacher des profils transversaux non symétriques, il est nécessaire d'avoir une procédure d'orientation (qui renvoie si on est à gauche ou à droite de la courbe grâce à la distance signée).

connaître l'orientation du point $\mathbf{p}$ par rapport à la courbe Γ_i puisqu'on souhaite utiliser des profils transversaux asymétriques : une courbe quadratique ne pouvant pas avoir de point d'inflexion, pour savoir si le point $\mathbf{p}$ est à gauche ou à droite de la courbe, il suffit de tester l'orientation du point $\mathbf{p}$ par rapport à la droite définie par Γ_i^0 et $\mathbf{q}$.

L'utilisation de courbes décomposées en morceaux de cubiques (c'est-à-dire de degré 3) qu'elles soient des courbes de Bézier ou d'Hermite permet d'obtenir une continuité C^2. Ces courbes sont plus difficile à manipuler : il n'existe pas, par exemple, de formule analytique pour calculer la distance entre un point et une de ces courbes [304]. Possédant potentiellement un point d'inflexion, la procédure pour décider de l'orientation nécessite de calculer la projection d'un point sur la courbe, de trouver la paramétrisation de cette projection selon l'abscisse curviligne, et de calculer la tangente au point projeté.

D'autres types de courbes peuvent être utilisées. Il est possible manipuler des arcs de cercles qui, quand ils sont assemblés, peuvent construire des trajectoires G^1. Ces dernières sont très utilisées pour représenter des routes. Les opérations de calcul de distance, d'orientation ou même de décalage (*offset*) sont relativement simples à exprimer avec des arcs de cercles. Dans la réalité, on construit généralement les routes avec des courbes un peu plus complexes dont la courbure varie linéairement et qui sont nommées clothoïdes. Les trajectoires produites par les courbes composées d'arc de cercles ou de clothoïdes peuvent paraître artificielles pour construire des reliefs naturels, mais il est possible de déformer les fonctions de distances par des bruits. Il existe d'autres types de courbes qui présentent des avantages en terme de calculs et de continuité comme les Splines Circulaires [356].

Quel que soit le type de courbe, on note $d(\mathbf{p}) = ||\mathbf{p} - \mathbf{q}||$ la distance signée entre $\mathbf{p}$ et la courbe Γ et $\mathbf{q}$ la projection de $\mathbf{p}$ sur la courbe Γ. Soit c le profil transversal d'élévation passant par $\mathbf{q}$, construit par interpolation en utilisant les coordonnées curvilignes sur la courbe Γ. De la même

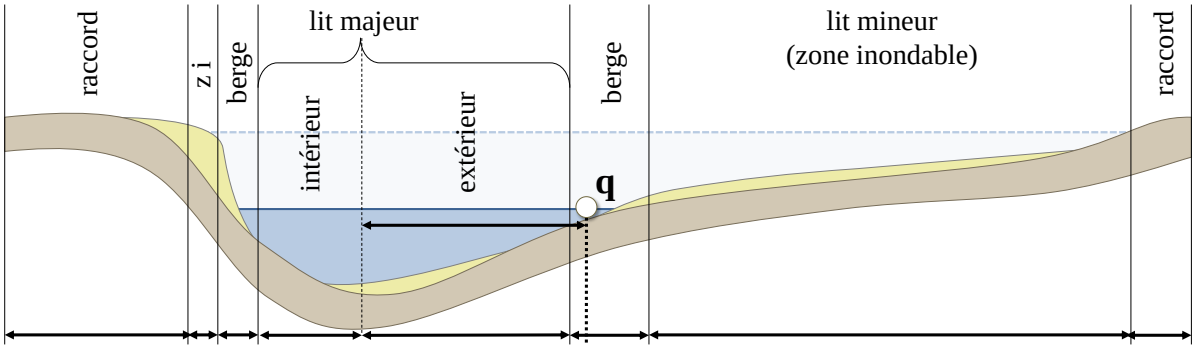


FIGURE 52 : Les profils de rivières sont construits de manière asymétrique : la courbe correspondant à la trajectoire passe par le point **q**. Le lit majeur (c'est-à-dire le niveau d'eau habituel) peut être excentré. La zone de raccord (qui correspond au moment où la fonction de poids décroît) permet de mélanger la rivière avec son environnement.

manière, c_α est le profil de poids correspondant, également construit par interpolation. Nous définissons :

$$f(\mathbf{p}) = \mathbf{q}_z + c \circ d(\mathbf{p}), \qquad \alpha(\mathbf{p}) = c\_\alpha \circ d(\mathbf{p}).$$

Plusieurs types de modelés peuvent être modélisés à l'aide de primitives curvilignes. Les rivières et les routes sont représentées par des squelettes courbes. Ces primitives peuvent également être déformées à l'aide de bruits pour représenter des éléments naturels de plus grandes tailles comme des crêtes et des vallées.

Cas des rivières

Les rivières sont créées à l'aide de profils courbes qui définissent la trajectoire du cours d'eau Γ. Les profils (Fig. 52) permettent de représenter les plans de coupe de la rivière en fonction de sa forme. Les rivières ont de nombreux types de profils et de trajectoires. Les coupes transversales les plus complexes peuvent également représenter des profils multiples : on parle alors de rivières en tresse. Pour plus d'informations sur les types de rivières existant dans la nature, nous renvoyons à la classification de Rosgen présentée en Section 5.1.2.

Entre deux profils transversaux, le relief est interpolé selon une fonction cubique. Cette interpolation est également utilisée pour calculer la hauteur de l'eau. Les zones de raccords correspondent aux endroits où la fonction de poids diminue progressivement pour permettre à la rivière de s'intégrer dans un relief. Si la zone de raccord est étroite et que le squelette de la primitive se trouve en dessous du terrain où l'on souhaite intégrer la rivière, il est possible de modéliser des canyons Fig. 46.

Les confluences (c'est-à-dire l'endroit où une rivière se jette dans une autre) nécessitent des profils paramétrés qui sont difficiles à mettre en place. La plupart des jonctions et des tresses sont modélisées à l'aide de primitives-images qui subissent une déformation géométrique et dont le relief est enrichi avec des fonctions de bruits. Pour obtenir ces phénomènes complexes, il est aussi possible d'utiliser des opérateurs adaptés qui mélangent plusieurs rivières existantes comme expliqué dans la section suivante.

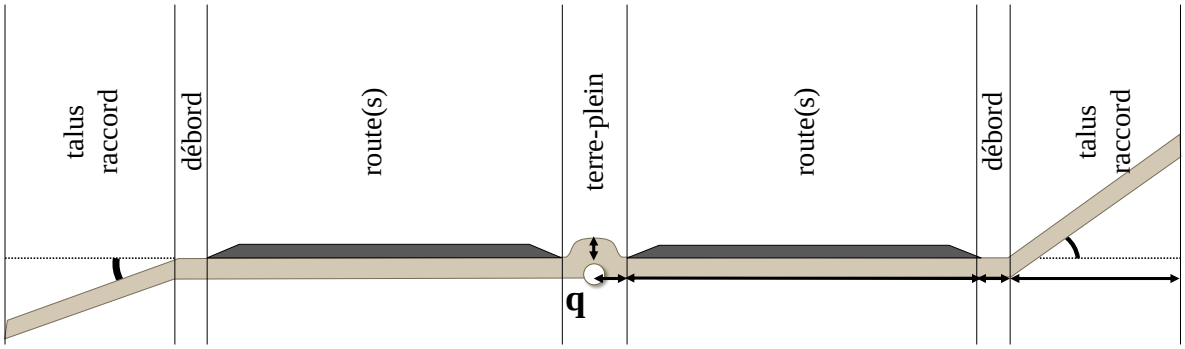


FIGURE 53 : Les profils de routes sont plus simples et peuvent être paramétrés par le nombre de voies et la présence ou non d'un terre-plein. L'angle formé par le talus peut être neutre, positif (la route sera alors encaissée) ou négatif (la route sera alors surélevée). Le même type de profil peut être utilisé pour les trajectoires correspondant aux chemins de fer.

Cas des routes

Le design des routes est un domaine qui est très réglementé et que les ingénieurs ont beaucoup étudié. Les trajectoires et les profils respectent des standards nationaux qui dépendent souvent des limites de vitesse en vigueur et du relief local du paysage (pente, type de sol) mais aussi de certains aspects visuels (obstacles qui empêchent une bonne visibilité, qualité des panoramas). Les profils transversaux sont paramétrés par le nombre de voies et par leur taille individuelle. Les routes sont aussi généralement légèrement inclinées pour permettre l'évacuation de l'eau de pluie. Une présentation générale de ces problématiques peut être trouvée sur Internet [78].

Les primitives de routes reposent sur le même modèle que les rivières mais avec des plans de coupe différents (Fig. 53). Le squelette est une fonction continue C^1 définissant généralement des morceaux de clothoïdes ou d'arcs de cercles qui représentent la trajectoire de la route. Les profils définissent les plans de coupe de la route qui sont paramétrés par la taille de la route et des débords. Ces primitives définissent aussi l'élévation dans un proche voisinage de la route pour permettre une intégration propre avec d'autres morceaux de terrain.

La plupart du temps, il est également possible d'approximer des trajectoires courbes avec des lignes polygonales échantillonnées avec précision. L'ensemble des algorithmes d'orientation, de distance ou de décalage et qui touchent les segments, les arcs de cercles et les courbes de Bézier peut être retrouvé dans le cours de PATEL [280].

Paramétrisation des courbes

Nous proposons également une paramétrisation le long des courbes qui permet de construire des rivières et des confluences complexes (Fig. 54). En pratique, cela nous permet de projeter des cartes d'élévation le long de ces courbes. Dans notre implémentation, les rivières, qui comportent des variations d'élévation le long de leur trajectoire, sont construites en connectant plusieurs primitives-courbes à images. Les différentes images définissant l'élévation et le poids se raccordent sur les bords de manière continue.

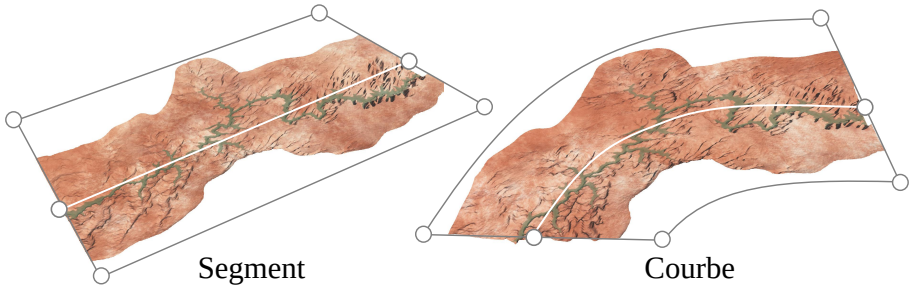


FIGURE 54 : Utilisation de données échantillonnées (*Grand Canyon*, U.S.A.) le long de courbes. Grâce aux déformations et à l'environnement immédiat dans lequel va s'intégrer ce relief, un même morceau de terrain peut être utilisé plusieurs fois sans que l'on aperçoive une répétition de motif.

3.2.4 Les primitives quadrangulaires

Ces primitives sont créées à partir de quatre points de contrôle définissant un quadrangle convexe dans lequel le terrain sera généré. Un rayon d'influence r permet de définir la manière de décroître de la fonction de poids. Ces primitives nous permettent d'intégrer dans nos scènes des parcelles de terrains obtenus à partir de données réelles comme les montagnes présentées dans la Fig. 55.

Des versions plus simples permettant de représenter des rectangles et des carrées sont également disponibles dans le modèle de représentation. Le champ de distance produit par un quadrangle produit un quadrangle aux coins arrondis.

Dans notre implémentation, nous utilisons une paramétrisation sur des quadrangles convexes. Cette dernière découle d'un calcul d'interpolation bilinéaire inverse (c'est-à-dire étant donné un point dans le polygone, il s'agit de trouver quelle est sa paramétrisation (u,v)). Le détail de cet algorithme est disponible sur la page de QUÍLEZ [315].

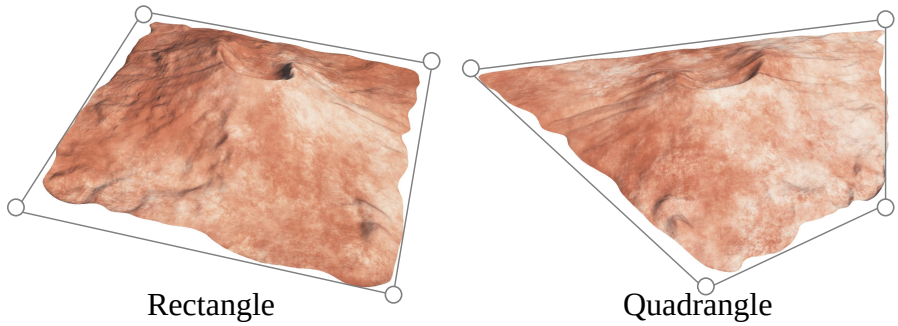


FIGURE 55 : Utilisation de données réelles (*Mont St. Helens*, U.S.A.) et projection sur deux quadrangles convexes. Pour un point donné dans le quadrangle, nous calculons ses coordonnées (u,v) permettant d'interpoler l'élévation à partir de la carte d'élévations.

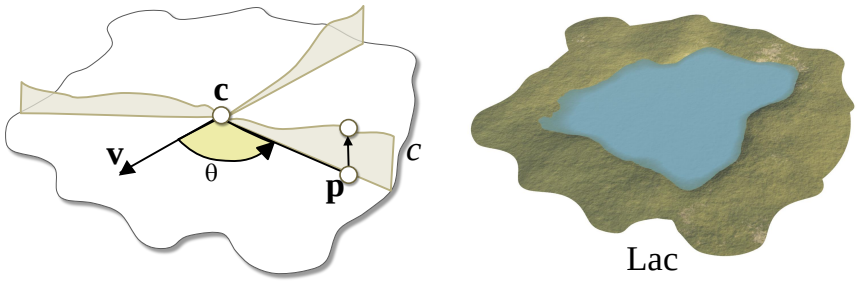


FIGURE 56 : Des formes complexes peuvent être définies à l'aide de ces formes polygonales. Un lac est créé en définissant un ensemble de profils croissants de manière radiale depuis le centre **c** de la forme.

3.2.5 Les primitives à contours

Ces primitives sont créées depuis une courbe γ étoilée autour d'un centre **c** et avec un ensemble de fonctions de profil $\{c_i\}$ associées à des angles θ_i qui décrivent l'élévation de manière radiale. Chaque section de coupe est stockée sous la forme d'une fonction uni-dimensionnelle par morceaux de quadriques (Fig. 56). La fonction d'élévation f est définie comme suit : soit un point **p**, nous calculons ses coordonnées polaires c'est-à-dire la distance $d(\mathbf{p})$ et l'angle $\theta(\mathbf{p})$ par rapport au centre **c** de la primitive et à son orientation **v**. Soit $\theta_i < \theta_\mathbf{p} < \theta_{i+1}$, on calcule :

$$t = \frac{\theta_\mathbf{p} - \theta_i}{\theta_{i+1} - \theta_i}$$

L'élévation d'un point $c(\mathbf{p})$ dépend de la section de coupe c. Cette coupe est construite à l'aide d'un ensemble de sections de coupe stockées explicitement que l'on interpole en fonction du secteur angulaire. Cela correspond à :

$$c(d) = \left(1 - (3t^2 - 2t^3)\right)\, c_i(d) + (3t^2 - 2t^3)\, c_{i+1}(d)$$

Les fonctions d'élévation et de poids de la primitive à contour sont définies par :

$$f(\mathbf{p}) = \mathbf{c}_z + c \circ d(\mathbf{p}), \qquad \alpha(\mathbf{p}) = c_\alpha \circ d(\mathbf{p}).$$

Ces primitives ressemblent aux fonctions de distance anisotropes des *implicit sweep objects* proposées par CRESPIN *et al.* [83].

3.3 Opérateurs

Les opérateurs sont des nœuds internes qui combinent les fonctions d'élévation et de poids de leurs sous-arbres pour définir deux nouvelles fonctions f et α. Nous considérons par la suite des nœuds binaires et notons les deux sous-arbres A et B.

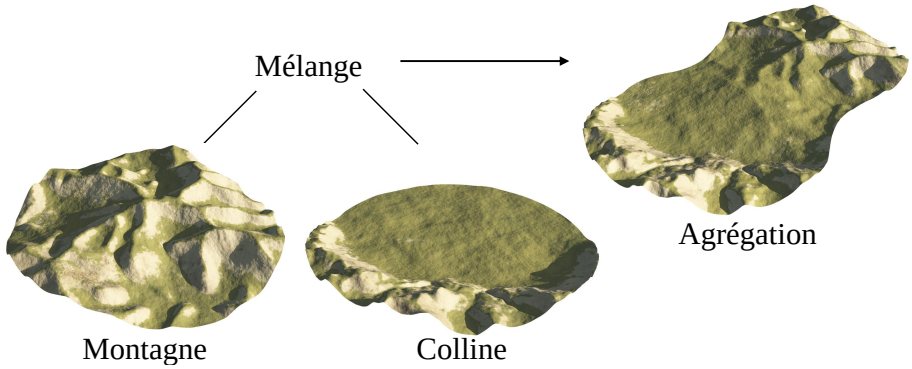


FIGURE 57 : L'opérateur de mélange permet d'agréger plusieurs primitives de manière continue. Ce nœud peut être transformé en un opérateur n-aire pour gérer plusieurs primitives.

3.3.1 L'opérateur de mélange

L'opérateur de mélange est un moyen efficace de créer de grands terrains à l'aide d'un ensemble de patchs. Le mélange de deux nœuds A et B combine les fonctions d'élévations f_A et f_B en fonction des poids α_A et α_B et permet d'agréger deux primitives de terrains (Fig. 57).

$$f = \frac{\alpha_A f_A + \alpha_B f_B}{\alpha_A + \alpha_B}, \qquad \alpha = \alpha_A + \alpha_B .$$

Quand deux primitives sont suffisamment éloignées (leurs supports Ω_A et Ω_B ne s'intersectent pas), elles n'ont pas d'influence l'une sur l'autre et le résultat est alors l'union des deux patchs de terrains produits par les sous-arbres A et B. Quand les primitives se rapprochent, les régions d'influence s'intersectent et les primitives se mélangent progressivement (Fig. 58). La zone de transition $\Omega_0 \setminus \Omega_T$ n'étant pas toujours visualisée, il est important de savoir qu'elle est également modifiée par cet opérateur. Il est possible de voir apparaître des zones non-connexes dans Ω_T.

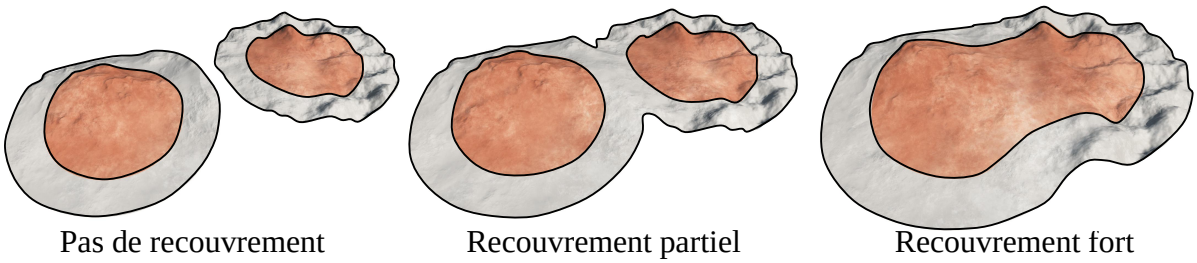


FIGURE 58 : Le mélange de deux primitives permet d'étendre le domaine de définition de notre terrain. Si les deux primitives sont disjointes, l'opérateur correspond à une union. Quand les primitives se rapprochent, leurs supports et leurs élévations se mélangent de façon continue.

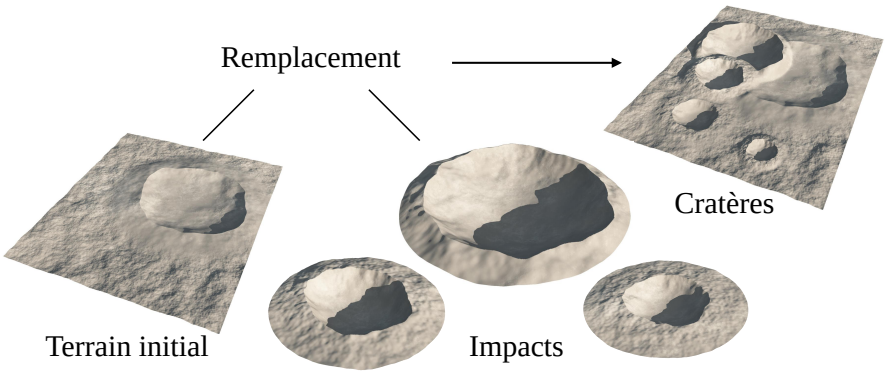


FIGURE 59 : L'opérateur de remplacement permet de sculpter facilement la forme d'un paysage. Il permet également de représenter une notion de temporalité : ici, le dernier opérande correspond au cratère le plus récent.

L'opérateur de mélange peut être facilement étendu pour devenir un opérateur n-aire. Cette propriété est importante et nous permet d'optimiser la structure de l'arbre quand beaucoup de primitives doivent se mélanger ensemble.

$$f = \sum_1^n f_i \alpha_i \Big/ \sum_1^n \alpha_i \,, \qquad \alpha = \sum_1^n \alpha_i \,.$$

L'opérateur de mélange étant associatif et commutatif, il est possible de reformuler hiérarchiquement un arbre de construction :

$$\mathcal{B}(A,B,C,D) = \mathcal{B}(\mathcal{B}(A,B),\mathcal{B}(C,D))$$

Cet opérateur, combiné à une grille accélératrice, permet de mélanger plusieurs dizaines de milliers de primitives de façon rapide et efficace.

3.3.2 L'opérateur de remplacement

Cet opérateur permet de remplacer un morceau de terrain dans un sous-domaine. On l'utilise pour placer des lacs, des monts, des rivières ou encore des routes sur un terrain existant. Ce nœud permet de remplacer de façon continue le sous-arbre gauche A par le sous-arbre B. Il peut être utilisé pour placer une forme géométrique précise contenant de l'eau sur un terrain comme un torrent ou un lac. Il est également facile de sculpter des paysages ayant subi des modifications temporellement plus récentes (terrassement d'un paysage naturel le long d'une trajectoire pour construire une route, impact d'un nouveau cratère sur une surface lunaire) (Fig. 59).

L'opérateur est asymétrique [3] : le nœud A est remplacé par B uniquement là où $\alpha_B > 0$:

$$f = (1 - \alpha_B) f_A + \alpha_B f_B \,, \qquad \alpha = \alpha_A \,.$$

3. L'opérateur est non commutatif et associatif à gauche.

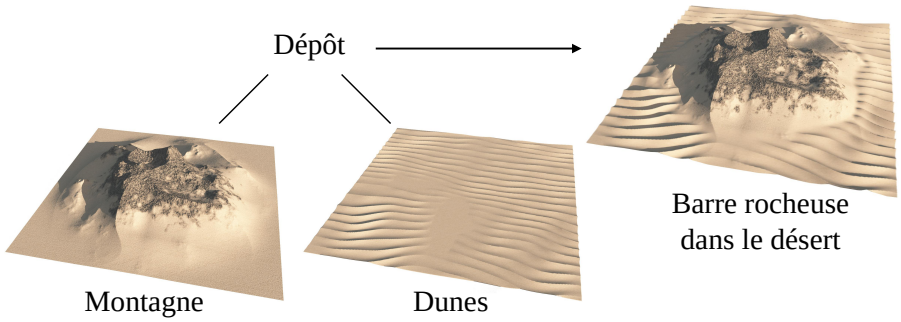


FIGURE 60 : Les dunes de sable ont été rajoutées à un paysage montagneux. Pour que le sable soit présent dans les basses altitudes sans recouvrir les roches, la fonction de poids α_B a été définie à l'aide de l'élévation f_A et du gradient ∇f_A de la montagne.

En général, nous voulons garder le poids de l'opérande gauche ce qui donne $\alpha = \alpha_A$. En fonction du contexte, nous pouvons également prendre en compte l'influence de la fonction de poids α_B et définir alors :

$$\alpha = (1 - \alpha_B)\,\alpha_A + {\alpha_B}^2,$$

ce qui remplace progressivement α_A par α_B.

3.3.3 L'opérateur de dépôt

Cet opérateur asymétrique[4]) ajoute localement des variations et des détails sur un terrain existant A en accord avec la fonction de poids α_B du deuxième opérande B (Fig. 60).

$$f = f_A + \alpha_B f_B, \qquad \alpha = \alpha_A.$$

De manière similaire à l'opérateur de remplacement, nous voulons souvent garder le poids de l'opérande gauche, ce qui revient à définir $\alpha = \alpha_A$. Une autre possibilité est de prendre en compte l'influence du deuxième argument α_B en définissant

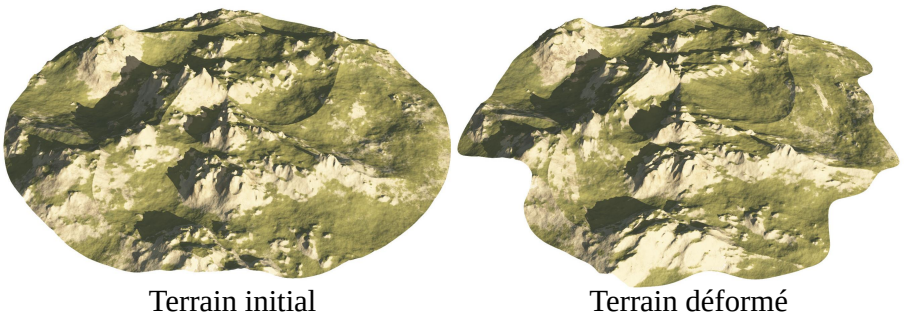
$$\alpha = \alpha_A + {\alpha_B}^2.$$

Ceci permet d'accroître le poids du résultat aux emplacements où se font les rajouts. Cet opérateur permet de rajouter des monticules ou des dunes sur un terrain relativement plat. Ce nœud peut se convertir facilement en un opérateur de retrait quand f représente une valeur négative.

3.3.4 L'opérateur de déformation

Cet opérateur permet une distorsion des parcelles de terrain en déformant le domaine spatial des fonctions d'élévation f et de poids α (Fig. 61). Une déformation est une fonction de classe C^2

4. L'opérateur est non commutatif et associatif à gauche.



Terrain initial Terrain déformé

FIGURE 61 : Il est possible de déformer aussi bien les primitives que des morceaux de terrains combinés. Une déformation permet d'éliminer la répétitivité que l'on peut retrouver quand on utilise massivement une même primitive.

bijective notée $\omega(\mathbf{p}) : \mathbf{R}^2 \rightarrow \mathbf{R}^2$. L'opérateur de déformation d'une primitive est défini comme un nœud unaire qui peut être évalué ainsi :

$$f(\mathbf{p}) = f_A \circ \omega^{-1}(\mathbf{p}), \qquad \alpha(\mathbf{p}) = \alpha_A \circ \omega^{-1}(\mathbf{p}).$$

Dans certains cas, nous voulons calculer le gradient de la fonction d'élévation, *e.g.*, pour évaluer une normale sur le terrain. Le gradient d'une fonction continue C^1 notée f, déformée par une fonction bijective ω est :

$$\nabla f(\mathbf{p}) = \nabla f_A \circ \omega^{-1}(\mathbf{p}) \times J_{\omega^{-1}}(\mathbf{p}).$$

Afin d'optimiser les temps de calcul, nous pré-calculons le Jacobien $J_{\omega^{-1}}$ pour chaque fonction de déformation ω présente sur le terrain.

3.3.5 L'opérateur de remplissage

L'opérateur de remplissage permet de calculer l'épaisseur nécessaire $f(\mathbf{p})$ pour combler un terrain f_A jusqu'à une hauteur donnée f_B. Cet opérateur est très utile pour remplir d'eau le lit des rivières (Fig. 62). Selon la fonction de poids α_B, cette hauteur sera ou non atteinte.

$$f(\mathbf{p}) = \begin{cases} 0 & \text{si } f_A \geq f_B, \\ (f_A - f_B) \times \max(1, \alpha_B) & \text{sinon}, \end{cases} \qquad \alpha(\mathbf{p}) = \alpha_B(\mathbf{p}).$$

3.4 Instanciation et opérateurs de transformations

Chaque morceau de terrain peut être décrit dans le repère du monde. Afin de permettre la réutilisation d'une parcelle de terrain déjà existante ailleurs, nous proposons de transformer

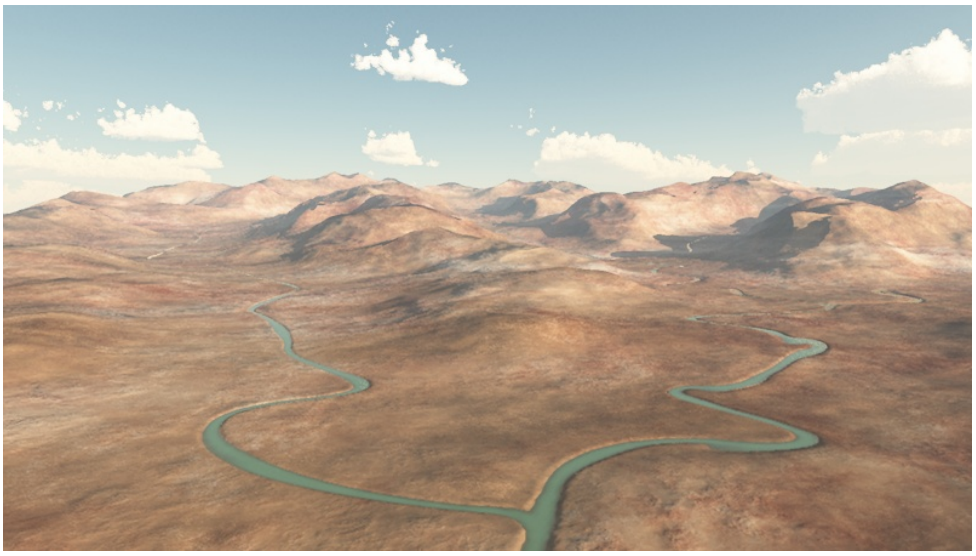


FIGURE 62 : L'opérateur de remplissage permet de facilement construire l'épaisseur de la couche d'eau d'une rivière (pour obtenir une élévation souhaitée), y compris quand cette couche est défini sur un terrain en pente.

l'arbre de construction en un graphe orienté acyclique. Nous définissons dans le modèle la notion d'instance : c'est un morceau de terrain défini par une primitive ou un arbre de construction qui est utilisé plusieurs fois dans une scène. Ces instances nécessitent donc des opérateurs de placement associés (Fig. 63). Chaque instance peut être elle-même déformée de façon unique. Dans le cas où une parcelle de terrain est instanciée plusieurs fois sans déformation, elle ne sculptera pas le terrain de manière identique. En effet, chaque instance sera combinée dans un environnement différent ce qui créera un résultat unique à chaque fois.

Les transformations affines sont des opérateurs de déformation spécifiques. Bien que le résultat revienne à appliquer les mêmes transformations sur les squelettes des primitives, l'avantage vient de la possibilité de transformer un morceau de terrain complexe correspondant à tout un sous-arbre (*i. e.* plusieurs primitives). Ainsi, chaque primitive est définie dans un repère propre et avec une orientation et une mise à l'échelle canoniques.

Nous détaillons maintenant les opérations de base qui correspondent à des transformations affines.

Translations

Un nœud de translation est un opérateur unaire qui stocke un vecteur de transformation $\mathbf{t}$ à la création et qui s'écrit sous la forme :

$$f(\mathbf{p}) = f_A(\mathbf{p} - \mathbf{t}), \qquad \alpha(\mathbf{p}) = \alpha_A(\mathbf{p} - \mathbf{t}). \qquad \mathbf{t} = \begin{pmatrix} t_x \\ t_y \end{pmatrix}.$$

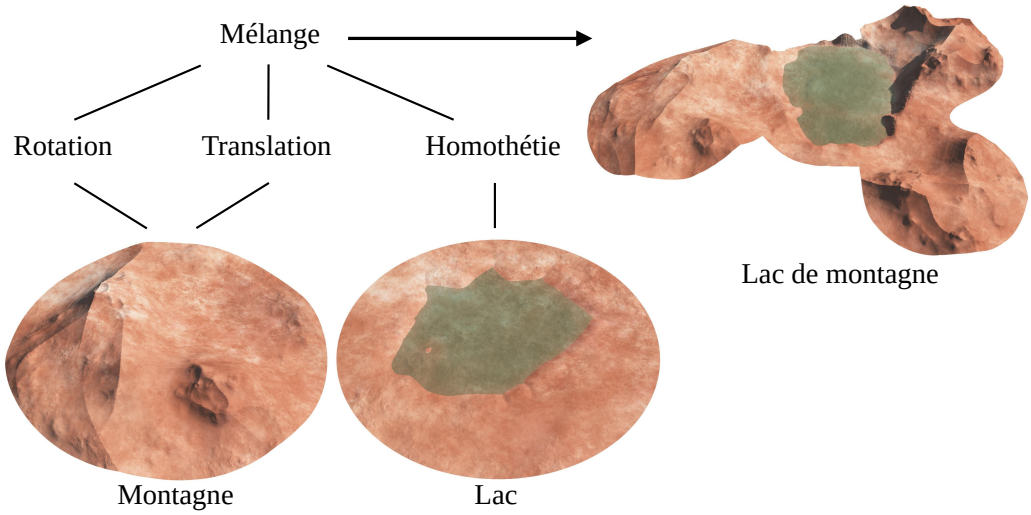


FIGURE 63 : L'instanciation associée aux opérateurs de transformations (translation, rotation, homothétie) permet de réutiliser une primitive plusieurs fois dans une scène.

Rotations

Un nœud de rotation de centre $\mathbf{c}$ est un opérateur unaire qui stocke un angle (en radian) θ à la création et qui s'écrit sous la forme :

$$f(\mathbf{p}) = f_A(\mathbf{R}_0 \cdot \mathbf{p} + \mathbf{t}), \qquad \alpha(\mathbf{p}) = \alpha_A(\mathbf{R}_0 \cdot \mathbf{p} + \mathbf{t}),$$
$$\mathbf{R}_0 = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \qquad \mathbf{t} = \begin{pmatrix} -\mathbf{c}_x \cos\theta + \mathbf{c}_y \sin\theta + \mathbf{c}_x \\ -\mathbf{c}_y \sin\theta - \mathbf{c}_y \cos\theta + \mathbf{c}_y \end{pmatrix}.$$

Homothétie

Un nœud de mise à l'échelle est un opérateur unaire qui stocke un point de référence $\mathbf{c}$ (centre de l'homothétie) et un scalaire de mise à l'échelle s. Les fonctions correspondantes s'écrivent :

$$f(\mathbf{p}) = f_A(\mathbf{S}_0 \cdot \mathbf{p} + \mathbf{t}), \qquad \alpha(\mathbf{p}) = \alpha_A(\mathbf{S}_0 \cdot \mathbf{p} + \mathbf{t}),$$
$$\mathbf{S}_0 = \begin{pmatrix} s & 0 \\ 0 & s \end{pmatrix}, \qquad \mathbf{t} = \begin{pmatrix} -s \cdot \mathbf{c}_x + \mathbf{c}_x \\ -s \cdot \mathbf{c}_y + \mathbf{c}_y \end{pmatrix}.$$

Quand $s = -1$, on a une symétrie de centre $\mathbf{c}$. En différenciant le s selon les coordonnées x et y, il est possible de réaliser une mise à l'échelle non uniforme (pour aplatir une primitive selon un des deux axes).

Symétries axiales

Un nœud de symétrie axiale est un opérateur unaire qui réalise la réflexion d'un point par rapport à une droite définie par un point $\mathbf{c}$ et un vecteur directeur unitaire $\mathbf{a}$. Les fonctions correspondantes s'écrivent :

$$f(\mathbf{p}) = f_A(\mathbf{S}_a \cdot \mathbf{p} + \mathbf{t}), \qquad \alpha(\mathbf{p}) = \alpha_A(\mathbf{S}_a \cdot \mathbf{p} + \mathbf{t}),$$
$$\mathbf{S}_a = \begin{pmatrix} {a_x}^2 - {a_y}^2 & 2 \cdot a_x \cdot a_y \\ 2 \cdot a_x \cdot a_y & {a_y}^2 - {a_x}^2 \end{pmatrix}, \qquad \mathbf{t} = \begin{pmatrix} -{a_x}^2\,\mathbf{c}_x + {a_y}^2\,\mathbf{c}_x - 2\,\mathbf{c}_y\,a_x\,a_y + \mathbf{c}_x \\ -{a_y}^2\,\mathbf{c}_y + {a_x}^2\,\mathbf{c}_y - 2\,\mathbf{c}_x\,a_x\,a_y + \mathbf{c}_y \end{pmatrix}.$$

3.5 Niveaux de détails

Le modèle d'arbre de construction permet de représenter une fonction d'élévation continue. Cette fonction d'élévation peut comporter des détails de hautes fréquences qui ne sont visibles qu'à très courte distance. On cherche à adapter le calcul de f en utilisant un système de niveau de détails. Ce système repose soit sur des nœuds internes décidant d'un niveau de détails, soit sur des primitives spécifiques paramétrées par un niveau de détails. Nous notons

$$\kappa(\mathbf{p}) \;:\; \mathbf{R}^2 \to [0,1]$$

une fonction continue C^2 définissant le niveau de détails, voulu à une instant donné, pour tout point $\mathbf{p}$. Quand $\kappa(\mathbf{p}) = 0$, on est au niveau de détails minimum et quand $\kappa(\mathbf{p}) = 1$, on est au niveau de détails maximum.

3.5.1 Opérateurs de changement de niveaux de détails

Les opérateurs de changement de niveaux de détails sont des nœuds binaires dont les sous-arbres sont évalués en fonction du niveau de détails voulu en entrée $\kappa(\mathbf{p})$. Ils s'inspirent des nœuds de niveaux de détails proposés par Barbier *et al.* [17]. Ces nœuds sont caractérisés par deux valeurs faisant office de seuil : k_A et k_B avec $0 \leq k_A < k_B \leq 1$ qui définissent quel sous-arbre doit être évalué en fonction du niveau de détails voulu. La fonction d'élévation est définie comme suit :

$$f(\mathbf{p}) = \begin{cases} f_A & \text{si } \kappa(\mathbf{p}) \leq k_A, \\ (1-t)\,f_A + t\,f_B & \text{si } k_A < \kappa(\mathbf{p}) \leq k_B, \\ f_B & \text{sinon}, \end{cases} \qquad \text{avec} \quad t = \frac{\kappa(\mathbf{p}) - k_A}{k_B - k_A}.$$

La fonction de poids α est calculée de la même manière.

$$\alpha(\mathbf{p}) = \begin{cases} \alpha_A & \text{si } \kappa(\mathbf{p}) \leq k_A, \\ (1-t)\,\alpha_A + t\,\alpha_B & \text{si } k_A < \kappa(\mathbf{p}) \leq k_B, \\ \alpha_B & \text{sinon}. \end{cases}$$

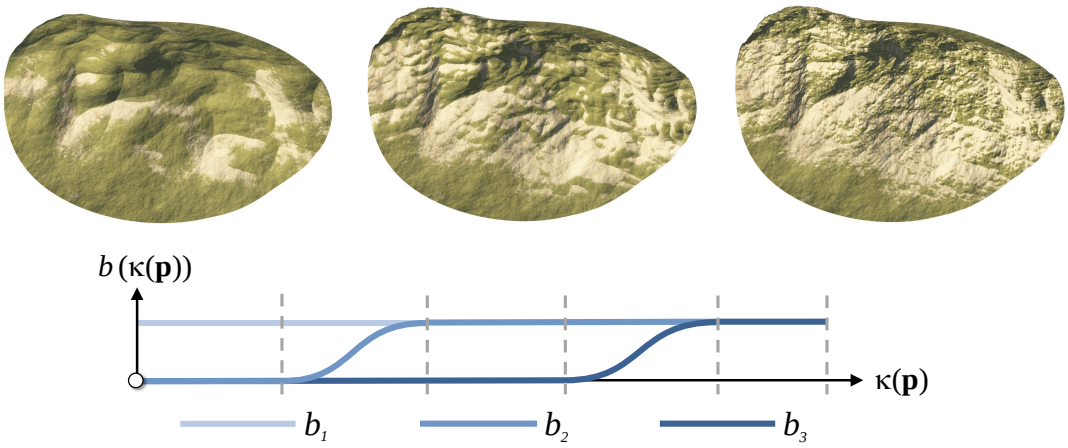


FIGURE 64 : Un exemple de primitive-disque paramétrée par un niveau de détails. Alors que le niveau de détails global κ augmente, les fonctions b_i vont progressivement croître pour ajouter du détail.

3.5.2 Les primitives à niveaux de détails continus

Ces primitives sont caractérisées par une fonction d'élévation $f(\mathbf{p})$ dont le calcul dépend de la fonction de niveaux de détails $\kappa(\mathbf{p})$. Ces primitives à niveaux de détails simplifient automatiquement leur évaluation dans l'objectif d'accélérer les calculs quand le niveau de détail diminue.

Notre système implémente plusieurs types de primitives à niveaux de détails continus. Un exemple est la primitive disque dont la fonction d'élévation correspond à une somme de fonctions de bruits d'amplitudes décroissantes et de fréquences croissantes. Quand cette primitive est interrogée avec un faible niveau de détails, elle ne somme que les plus basses fréquences. Entre deux niveaux de détails, elle interpole les élévations afin d'obtenir une transition douce et continue. Nous définissons $f(\mathbf{p})$:

$$f(\mathbf{p}) = \mathbf{c}_z + \sum_{i=1}^{h} a_i \, \mathrm{n}\big((\mathbf{p}-\mathbf{c})\, s_i\big) b_i \circ \kappa(\mathbf{p}) \,.$$

Les fonctions de filtres $b_i : [0,1] \to [0,1]$ sont définies comme des fonctions augmentant doucement le niveau de détails $\kappa(\mathbf{p})$ sur l'intervalle $[k_i^-, k_i^+]$ (Fig. 64). Nous avons toujours $b_i(1) = 1$ (si on demande le maximum de détails, on somme toutes les harmoniques). De même $b_0(0) = 1$: au niveau de détails le plus faible, on calcule tout de même l'harmonique de plus basse fréquence. Soit $\kappa_\mathbf{p} = \kappa(\mathbf{p})$, le niveau de détail global souhaité.

$$b_i(\kappa_\mathbf{p}) = \begin{cases} 0 & \text{si } \kappa_\mathbf{p} \le k_i^- \,, \\ \left(1-t^2\right)^3 & \text{si } k_i^- < \kappa_\mathbf{p} \le k_i^+ \,, \\ 1 & \text{sinon} \,, \end{cases} \qquad \text{avec} \quad t = \frac{\kappa_\mathbf{p} - k_i^-}{k_i^+ - k_i^-} \,.$$

Le niveau des détails contenus peut être contrôlé par la définition des différents intervalles $[k_i^-, k_i^+]$. Dans notre implémentation, nous définissons $\kappa(\mathbf{p})$ de manière à ce que le niveau de détails diminue alors que la distance entre la caméra et le point évalué augmente.

3.6 Gestion multi-couches

Les modèles multi-couches permettent de représenter des terrains comportant plusieurs matériaux différents (*e. g.* l'eau ou le sable). Il est possible de manipuler des morceaux de terrains qui auront chacun des propriétés visuelles et physiques différentes. L'intérêt est de pouvoir associer des matériaux et des textures propres à chaque type de sol (terre, roche, sable) et de permettre l'utilisation d'algorithmes qui manipulent la notion de matériaux. Des simulations montrent que l'érosion hydraulique ou les changements géologiques influencent différemment le terrain selon sa composition [29, 328]. Ainsi l'écoulement de l'eau affecte davantage les sols meubles et sédimentaires.

Si les simulations nécessitant des calculs de voisinages (sur des grilles par exemple) ne sont pas possibles dans le modèle de représentation proposé, il est toutefois possible d'imaginer des opérateurs (ou des outils) manipulant spécifiquement la notion de matériaux. Ainsi, l'opérateur de remplissage est particulièrement adapté pour définir l'épaisseur d'eau locale, le long d'une trajectoire correspondant à une rivière.

Modèle à plusieurs arbres

Pour représenter plusieurs matériaux, nous proposons de réutiliser le modèle de terrain vectoriel par arbre de construction pour définir des couches de matériaux homogènes. Nous représentons alors un terrain comme une liste ordonnée de couches, donc une liste d'arbres de construction (Fig. 65). L'ordre détermine la superposition des résultats.

Un des problèmes de cette représentation est l'apparition de discontinuités sur les bords du domaine. Si au bord du domaine d'existence d'une des couches supérieures (par exemple, du sable sur de la roche), l'épaisseur n'est pas nulle, alors, il existera une frontière discontinue.

Pour résoudre ce problème, il est possible d'utiliser un opérateur de raccord spécifique. L'opérateur de lissage multiplie la hauteur par le poids afin d'annuler progressivement l'épaisseur. Ajouter une couche de matériaux par dessus un terrain existant se fera donc avec un raccord continu : au bord du domaine de définition de cette couche, l'épaisseur sera nulle.

$$f(\mathbf{p}) = f_A\, \alpha_A\,, \qquad \alpha(\mathbf{p}) = \alpha_A\,.$$

Pour certaines couches de matériaux, il est intéressant de pouvoir définir l'épaisseur d'un matériau en fonction de l'altitude d'une autre couche : cela nécessite que les arbres de construction des couches supérieures puissent être construits en évaluant les arbres de construction des couches inférieures. Par exemple, une couche de sable peut être définie pour recouvrir les zones inférieures d'un lac (Fig. 66). De la même manière, on peut faire en sorte que le sable ne soit pas présent là où la pente du terrain est trop forte.

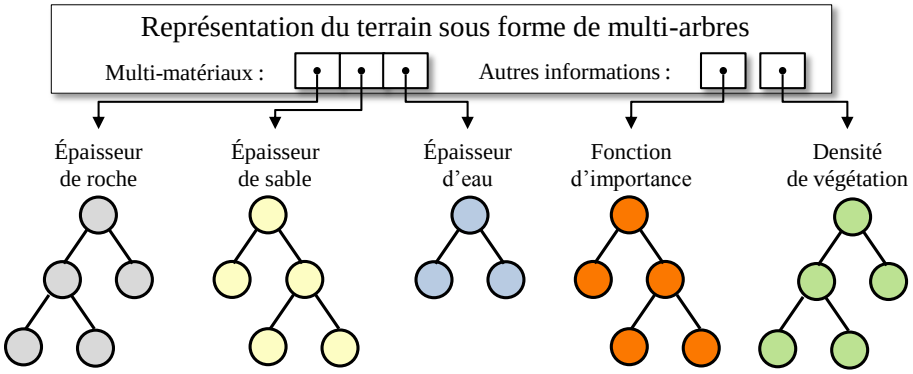


FIGURE 65 : Dans la plupart des figures de ce document, c'est la représentation multi-arbres qui est utilisée. Celle-ci a plusieurs avantages : elle est contrôlable et générique (tous les arbres utilisent les mêmes types de nœuds classiques) et permet d'utiliser autant de fonctions que l'on souhaite (qui peuvent correspondre à des épaisseurs, à des champs d'informations ou encore à des densités).

Cette représentation est générique et permet de définir un nombre quelconque de couches (Fig. 67). Il est également possible de représenter des informations supplémentaires comme des fonctions d'importance (pour la visualisation) ou de densité (pour la végétation). Ce modèle permet de contrôler chaque couche de matériaux indépendamment des autres. Il est également efficace puisque chaque arbre de construction maintient sa propre hiérarchie de boîtes englobantes ce qui permet d'éviter de nombreux calculs si la contribution d'une couche de matériaux est nulle sur de grandes zones. Comme la composition géologique de chaque couche est homogène, les frontières entre les différentes couches sont nettes. La représentation n'est donc pas adaptée à représenter des mélanges hétérogènes. C'est pourquoi, pour rendre les scènes réalistes, on utilise généralement l'élévation ou le gradient de la couche inférieure pour définir la nouvelle épaisseur.

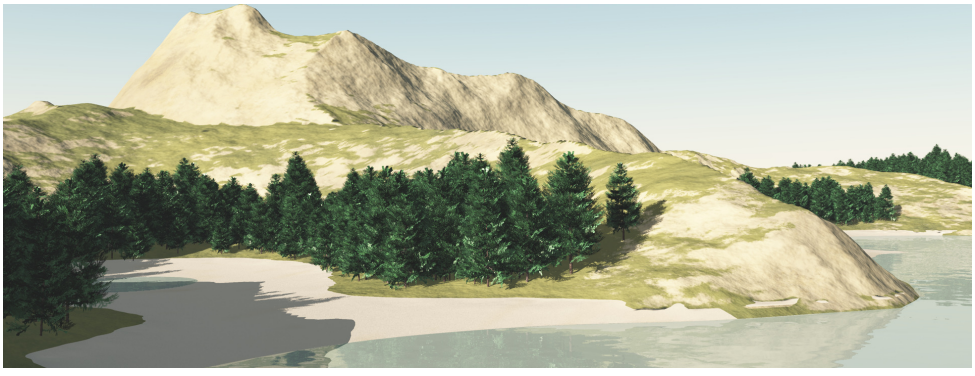


FIGURE 66 : Dans cette scène, le sable a été placé sur les pentes faibles, en dessous d'une certaine altitude. L'eau définit une autre couche de matériaux. Enfin, la densité de végétation est représentée par un arbre supplémentaire.

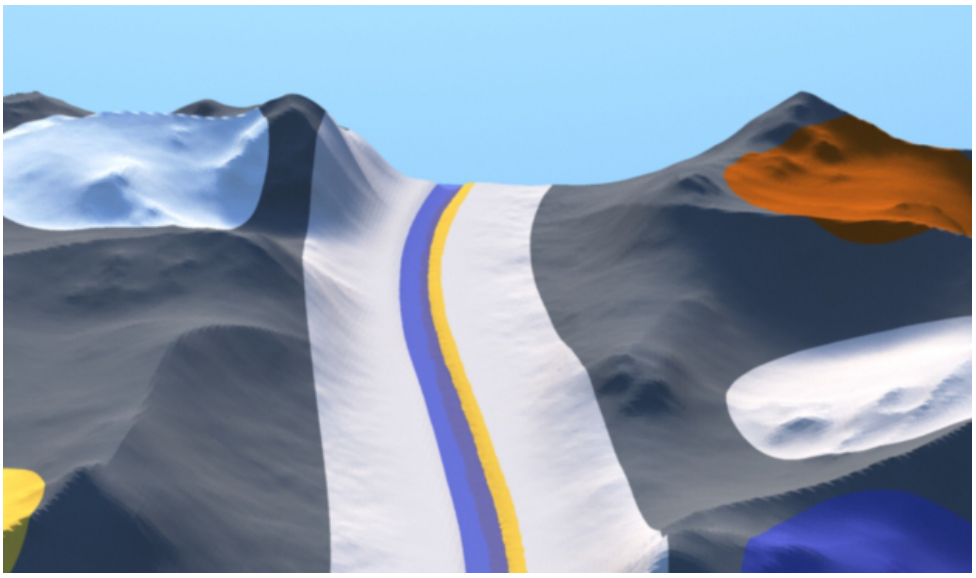


FIGURE 67 : Une scène de test d'utilisation de plusieurs couches de matériaux. Avec ce système, il est possible de définir autant de matériaux que l'on souhaite.

3.7 Résultats

Nous avons implanté cette représentation dans une bibliothèque écrite en C++. Tous les terrains présents en illustrations dans les chapitres 3, 4 et 5 sont représentés dans notre formalisme. Le relief a été extrait sous la forme d'un maillage surfacique à l'aide d'un ordinateur de bureau équipé d'un INTEL® Core *i*7 avec une fréquence de 3GHz et 16Go de RAM. Les images photoréalistes avec de la végétation ont été produites en texturant et en décorant chaque relief dans E-ON VUE XSTREAM®. Les scènes complètes ont ensuite été rendues en les important dans MENTALRAY®.

3.7.1 Performances mémoires

Les images vectorielles représentent les objets à l'aide de primitives géométriques (points, lignes, courbes, polygones) intuitives à manipuler. Ces structures ne stockent pas d'échantillons de l'objet (des pixels représentant une couleur ou une élévation). Ces représentations vectorielles permettent également de construire des scènes où les objets ont des résolutions ou des tailles très différentes. Comme nos primitives fonctionnent avec des supports compacts, nous pouvons considérer que le modèle de représentation est vectoriel.

Les modèles procéduraux représentent les objets à l'aide de formules mathématiques ou d'algorithmes. Ils sont très légers en mémoire (ils ne stockent pas l'objet, ils ne stockent même pas une description géométrique exacte de l'objet, mais ils stockent les paramètres de l'algorithme de construction de l'objet) et sont capables de générer des résultats visuellement complexes.

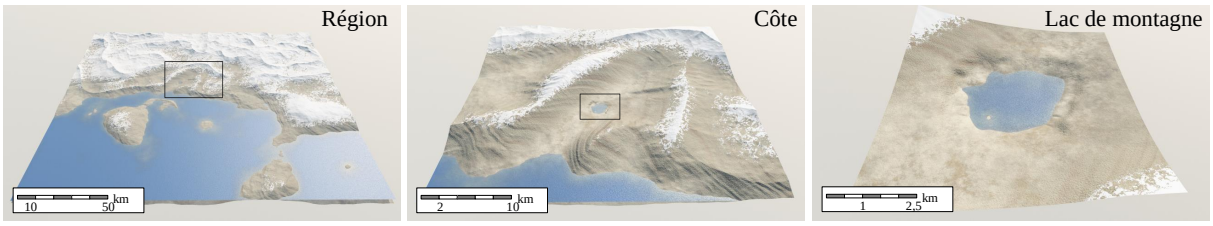


FIGURE 68 : Le modèle permet de représenter des scènes composées d'éléments de taille très variable. Dans cet exemple, le relief général a été généré en mélangeant plusieurs primitives-disques ; la montagne plus détaillée, qui borde la mer, provient d'une primitive à image. Enfin, le relief du lac a été créé par une combinaison de primitives disques et creusé par un opérateur de remplacement.

Malheureusement, ces représentations sont généralement peu intuitives. Une grande partie des fonctions d'élévation utilisables dans le modèle sont procédurales.

Le modèle de terrain mixe donc représentations vectorielles et représentations procédurales (Fig. 68). Chaque arbre de construction peut être stocké de manière compacte par un tableau de paramètres correspondant aux nœuds et à leurs paramètres. La structure est légère et nos expérimentations montrent (Tableau 2) que, dans la majorité des cas, une dizaine ou une centaine de primitives par kilomètre carré suffisent pour représenter les différents types de relief présents dans la nature (collines, montagnes, rivières), les détails au sol étant introduits par des fonctions de bruit. Des primitives supplémentaires pourraient encore enrichir le sol de détails avec des empreintes de pas, des traces de véhicules, des ridules sur le sable ou des crevasses.

Le système permet également de gérer de l'instanciation. Ce mécanisme autorise l'utilisation d'une même primitive, ou d'un même sous-arbre, de multiples fois et ce à plusieurs endroits. Chaque instance étant transformée de manière unique (mise à l'échelle, rotation), parfois déformée et enfin combinée dans son environnement propre, il n'y a généralement plus de similarité visuelle apparente. Un exemple de scène construite uniquement avec 3 types de primitives (la primitive de sable Fig. 60, la primitive de lac Fig. 56 et une trentaine d'instances de la primitive de montagne Fig. 61) est visible en Fig. 69.

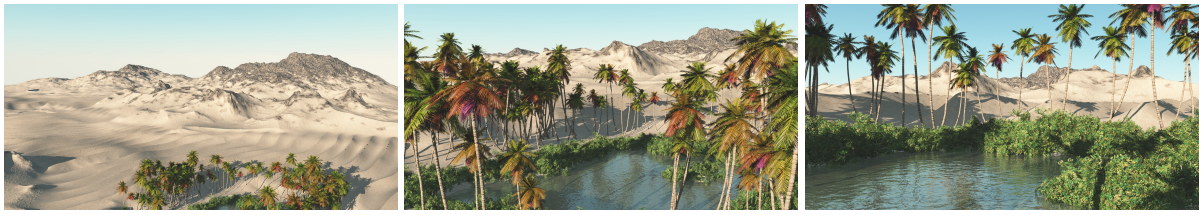


FIGURE 69 : Cette scène complexe est composée d'environ 30 instances d'une même et unique primitive de montagne (ce patch de relief est illustré en Fig. 61).

Tableau 2 : Statistiques mémoire pour la plupart des scènes. Le coût en mémoire est en ko, la superficie est exprimée en km^2. La plupart des scènes ont été créées en 15 à 55 minutes, la majeure partie de ce temps ayant été utilisée pour placer les détails. Le coût-mémoire des primitives-images utilisées dans la construction des scènes est inclus dans le calcul (par exemple, la primitive-image utilisée dans Fig. 68 est environ 61 ko).

Scène	Surface	Analyse mémoire		
	(km^2)	# Primitives	# Opérateurs	Coût-mémoire (ko)
Fig. 46	1×1	20	3	3
Fig. 68	200×200	15	3	81
Fig. 69	$x \times x$	x	x	x
Fig. 70	100×100	50	25	7
Fig. 71	2×2	78	10	12
Fig. 75	2×2	70	9	10
Fig. 87	0.6×0.6	148	9	19
Fig. 111	31×31	19826	503	1966
Fig. 114	50×50	48618	1218	4808
Fig. 115	55×55	73970	1531	7496
Fig. 116	56×56	81653	1609	8180
Fig. 117	40×40	31055	718	3058
Fig. 118	58×58	72731	1666	7174
Fig. 119	55×55	65670	1520	6362

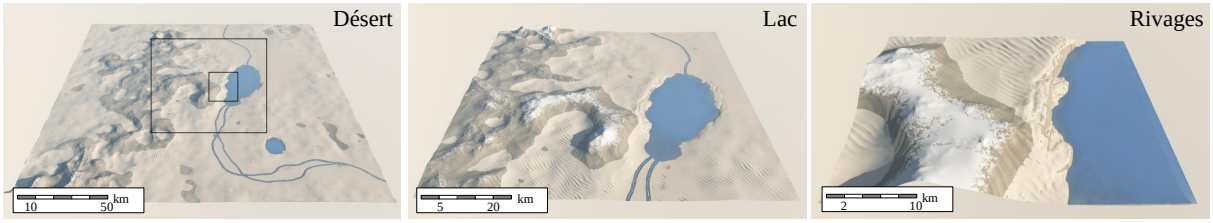


FIGURE 70 : Notre représentation, combinée avec des textures et un rendu photo-réaliste permet de créer des reliefs complexes. L'arbre de construction de cette scène n'est composé que de trois types de primitives : le lac correspond à la Fig. 56, les dunes de sables à la Fig. 60 et les montagnes sont plusieurs instances de la Fig. 61. L'épaisseur de la couche d'eau est définie à l'aide d'un arbre de construction supplémentaire.

3.7.2 Expressivité

Avec ce modèle (et la bibliothèque actuelle composée de modelés de collines, de montagnes ou encore de rivières), nous sommes capables de représenter la plupart des reliefs présents dans la nature tout en recouvrant la surface du relief de petits détails à l'aide de fonctions de bruit bien réglées. Nous pouvons également enrichir toutes nos scènes avec des primitives supplémentaires représentant de très petits détails au sol qui ne correspondent pas à du bruit comme des empreintes de pas ou de véhicules, des ridules sur le sable, la modification du terrain provoquée par les racines d'un arbre ou des crevasses.

Notre approche permet de construire et e manipuler des terrains complexes et variés en composant à la fois des primitives définissant des caractéristiques de grande taille (montagnes, plaines, plateaux) et des primitives de plus petite taille pour sculpter et éditer des détails de relief (rivières, dunes, lacs). On peut donc construire et éditer un terrain hiérarchiquement et de façon à ce qu'il comprenne de nombreux modelés de différentes résolutions (Fig. 70).

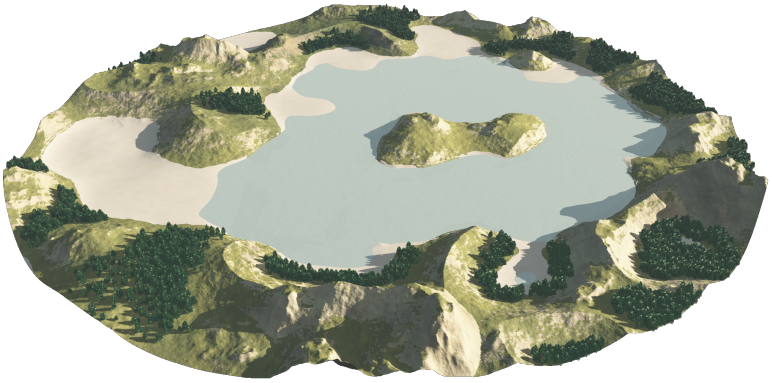
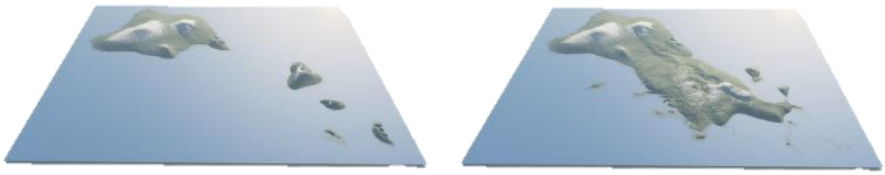


FIGURE 71 : Notre modèle permet de construire des scènes complexes. En utilisant des arbres de construction supplémentaires, il est possible de représenter plusieurs couches de matériaux (ici, la roche, le sable et l'eau). La densité de végétation peut être gérée de la même manière.



Données réelles Terrain édité

FIGURE 72 : Importation et modifications du relief de Hawaii dans notre formalisme.

Les mécanismes de niveaux de détails permettent de construire des terrains dont le relief peut se simplifier quand la caméra est éloignée. Ces outils permettent d'extraire des surfaces plus rapidement en ignorant une partie des calculs. La transition entre deux niveaux de détails est assurée de manière continue.

Ce modèle permet une représentation multi-matériaux des scènes. Les représentations précédentes [30, 299] (présentées en Section 2.2.3) reposent souvent sur une structure de voxels ou de piles de matières qui sont très coûteuses en mémoire. La possibilité de construire et d'éditer des couches de matériaux de façon vectorielle permet de représenter des scènes complexes (Fig. 71).

Le modèle peut s'étendre facilement par l'ajout de nouvelles primitives ou d'opérateurs qui répondront aux besoins spécifiques d'un utilisateur. En pratique, pour développer un nouveau nœud dans le modèle hiérarchique, il suffit d'intégrer une nouvelle classe définissant $f(\mathbf{p})$ et $\alpha(\mathbf{p})$ ce qui consiste à écrire une dizaine de lignes de code.

Notre représentation permet aussi d'utiliser des données réelles sous forme de cartes de hauteurs. Leur coût-mémoire est non-négligeable mais elles sont utiles pour représenter certains reliefs très complexes (comme des tronçons de rivières ou des confluences). Il est relativement facile d'intégrer des fonctions procédurales pour modifier ces reliefs réels afin de les intégrer dans une scène (grâce à des déformations) ou pour les enrichir avec des détails procéduraux (Fig. 72).

3.7.3 Édition

Notre paradigme de modélisation s'appuie sur la notion de modelés (*feature modeling*). Les modelés étant des unités de reliefs visuellement homogènes, il est relativement intuitif de les manipuler. Avec cette représentation, on manipule donc des éléments de plus haut niveau sans s'intéresser au calcul de la géométrie de manière exacte (Fig. 73). En cela, notre représentation s'inspire de la *Constructive Solid Geometry* : chaque morceau de terrain est paramétré par l'artiste qui a aussi un contrôle précis sur le placement des primitives.

Les algorithmes de génération procédurale de terrains peuvent s'appuyer sur ce type de modèle comme le démontre la méthode de génération d'îles présentée dans le Chapitre 5 (et dans les articles associés [139, 140]) et qui repose sur ce modèle de représentation. Ainsi, pour le développement de ces algorithmes, le programmeur se concentre principalement sur la distribution des primitives et sur leur paramétrage.

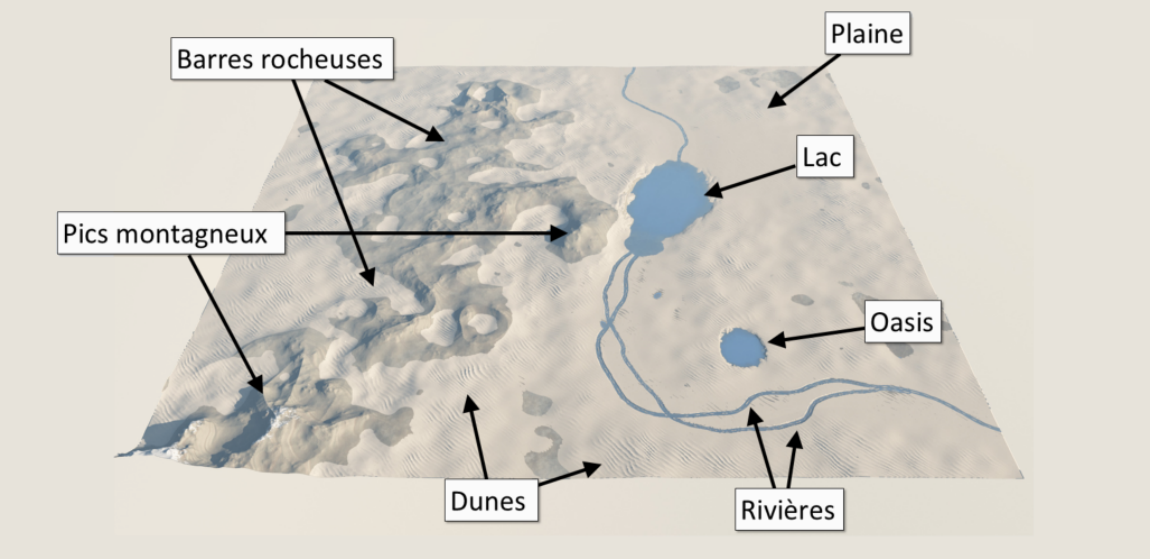


FIGURE 73 : La modélisation par modelés permet à l'infographiste de placer et de contrôler des éléments du relief indépendamment les uns des autres.

L'utilisation de primitives procédurales définies sur des domaines compacts et représentant différents types de structures géologiques/géographiques permet un contrôle à la fois local et semi-global. Il est facile pour un artiste d'éditer localement le terrain et d'augmenter les détails. Le modèle permet de construire des scènes de tailles variées (Fig. 68) avec un niveau de détails élevé (Fig. 71).

Une des principales clés de cette représentation est l'utilisation de fonctions de poids α à support compactes. Contrairement aux surfaces implicites classiques, chaque primitive différencie sa forme de son domaine. Comme les opérateurs manipulent à la fois les formes et les domaines, il est possible de construire des fonctions de combinaisons complexes mais pourtant intuitives. Les zones de transitions $\Omega_0 \setminus \Omega_T$ peuvent être visualisées et permettent un raccord continu du relief.

L'instanciation permet d'augmenter les performances du modèle, en ne représentant en mémoire qu'une seule et unique fois des modelés utilisés plusieurs fois dans la scène. En plus de cette optimisation en terme de mémoire, l'instanciation permet de proposer à l'utilisateur un outil de copier/coller très intuitif. Il est possible de sélectionner une sous-partie d'un arbre définissant un morceau de terrain et de la dupliquer en d'autres endroits.

La visualisation du modèle est également importante (elle est traitée dans le Chapitre 4), en particulier pour éviter des artefacts visuelles provenant de l'utilisation de fonctions multi-couches. Comme l'édition d'une scène dépend également de la possibilité d'interagir en temps-réel avec le modèle, de nombreuses optimisations sont mises en place pour accélérer l'évaluation de l'arbre de construction. En général, l'évaluation d'un arbre ne prend que quelques secondes sur une architecture CPU et seulement quelques millisecondes sur GPU. Le système de niveaux de détails, la hiérarchie de boîtes englobantes, l'élagage fréquentiel, la décomposition d'arbres

ou l'utilisation de structures accélératrices classiques (comme les *quadtrees*) sont autant d'optimisations facilement intégrables dans le modèle et qui influencent beaucoup les performances.

3.8 Limitations et perspectives

Nous présentons ici certaines limitations et perspectives qui touchent le modèle de représentation de terrains par arbres de construction.

3.8.1 Limitations

Notre modèle ne permet de représenter que des terrains 2D surfaciques sans surplombs, contrairement aux méthodes basées sur des piles de matières ou sur des voxels. Cela est dû à la nature des calculs d'évaluations dans l'arbre. Si cette contrainte **empêche la représentation de terrains réellement 3D** et nous fait perdre en expressivité, le modèle gagne en performances en proposant une évaluation dite *pure* (*i. e.* qui ne requiert de calculer la fonction qu'en un seul point). Il nous est toutefois possible de représenter plusieurs couches déposées les unes sur les autres à l'aide d'arbres supplémentaires. Le sable et l'eau des figure Fig. 69, Fig. 70 et Fig. 71 sont générés par deux arbres additionnels.

Nous avons proposé quelques primitives et opérateurs pour ce modèle de représentation. Dans une optique d'utilisation réelle du modèle, il serait **nécessaire d'enrichir notre bibliothèque avec de nouveaux types de nœuds** si l'on souhaite représenter de nouveaux reliefs. Cette opération est facile à effectuer mais il est tout de même nécessaire d'avoir une définition analytique des modelés que l'on veut représenter. Parmi les modelés que nous aimerions avoir dans notre bibliothèque figurent par exemple les mesas.

Comme mentionné ci-dessus, **la représentation interdit l'utilisation de simulations classiques** nécessitant des calculs de voisinage. Ainsi pour utiliser les méthodes d'érosion ou de *weathering* décrites dans l'état de l'art, il est nécessaire d'extraire un modèle 3D échantillonné depuis notre représentation fonctionnelle *pure*.

Si le modèle permet d'utiliser des données échantillonnées (comme des cartes de hauteurs), **ces primitives-images nécessitent de stocker des données** importantes dans l'arbre, ce qui impacte les performances-mémoire. Cette limitation est un défi majeur.

Enfin, nous proposons un nouveau paradigme de modélisation qui s'abstrait (ou en tout cas, s'éloigne) de la manipulation de la géométrie. Il présente des avantages par rapport aux outils classiques de modélisation et de subdivision mais un obstacle important à son adoption reste **l'inertie technologique et ergonomique** des infographistes et des développeurs.

3.8.2 Perspectives

Cette section présente les perspectives majeures de la représentation des terrains selon le modèle par arbre de construction.

Les techniques d'érosion habituelles utilisent principalement des modèles de simulations extrêmement coûteux en temps de calculs. Dans ces simulations, on retrouve une notion de voisinage : il n'est pas possible de simuler l'érosion en un point sans connaître l'état de l'environnement proche. De plus, ces simulations se calculent sur une discrétisation de l'environnement alors que le modèle par arbre de construction crée une fonction d'élévation continue. Malgré tout, on peut chercher à **modéliser une érosion ontogénétique fonctionnelle** (qui s'intéresse aux résultats visuels mais pas aux processus réels d'érosion). Un modèle d'érosion qui se prêterait à cette structure fonctionnelle (dont l'évaluation est *pure*) est proposée par DE CARPENTIER et BIDARRA [96]. Il repose sur la construction d'une fonction de bruits dont le domaine de définition est déformé à l'aide de son gradient.

La perspective la plus importante concerne la **modélisation procédurale inverse**. Si, à partir de données échantillonnées (un terrain réel ou construit par un artiste) il est possible de retrouver un arbre de construction de primitives qui approxime le relief de départ, alors ce modèle sera réellement complet et utilisable à une échelle industrielle. Cette perspective ouvre à elle seule de nombreuses applications : en premier lieu, ce modèle pourrait permettre la compression mémoire de terrains. Ensuite, il deviendrait possible d'utiliser des algorithmes de simulation d'érosion dans ce modèle : on passerait par une représentation échantillonnée pour faire les calculs. Enfin, de nombreux outils interactifs seraient utilisables par les infographistes pour éditer temporairement et localement des morceaux de terrains.

Une autre perspective concerne les textures et les matériaux. Actuellement, seul le relief grossier est modélisé dans la représentation. Pour obtenir un aspect visuel complet, il serait nécessaire de **pouvoir générer, plaquer et paramétrer des textures** tenant compte à la fois de la couleur et des micro-reliefs de la surface (au travers de *normals maps*). L'utilisation de textures procédurales paramétrables localement est une des solutions envisageables.

Les textures et matériaux en question devront être placés en fonction de la composition des sols. Actuellement, le modèle gère une notion de multi-matériaux sous forme d'un empilement de couches géologiquement homogènes. Un défi majeur consiste à **gérer plus finement les frontières entre ces matériaux**. Une interface peut correspondre à plusieurs cas de figures : nous les appelons frontières marquées (*e. g.* entre de la terre et de l'asphalte), transitions floues (*e. g.* entre des galets et du sable), et enfin transitions hétérogènes (*e. g.* entre de la terre et du sable).

FIGURE 74 : En 2007, le CryEngine 2 propose des outils pour mêler cartes de hauteurs et représentations non-planaires

La représentations des surplombs constitue un autre problème. Le formalisme que nous avons proposé permet de construire des fonctions $f(\mathbf{p}_x, \mathbf{p}_y) = z$. L'utilisation d'une fonction de poids associée à chaque nœud, couplée à la compacité des primitives permet de contrôler facilement la création d'un terrain planaire. Pour **construire des terrains avec des surplombs**, il est possible d'utiliser un formalisme connu de longue date comme celui des surfaces implicites où on construit une isosurface $f(\mathbf{p}_x, \mathbf{p}_y, \mathbf{p}_z) = 0$. Coupler les deux modèles est certainement possible mais il est nécessaire de définir proprement les interfaces entre les deux types de fonctions (afin de profiter autant que possible des performances et du contrôle du modèle planaire). L'utilisation de surfaces implicites sur des cartes de hauteurs a déjà été implantée dans certains moteurs graphiques (Fig. 74) comme le CryEngine 2 (des démonstrations sont disponibles sur Internet [87]) ou dans le système Arches [299].

Enfin, l'arbre de construction (ou le graphe acyclique orienté) représente un arbre syntaxique abstrait (AST) [5]. Il est possible d'**analyser cet AST pour le simplifier** afin de diminuer sa taille et ainsi optimiser les calculs d'évaluation. La réorganisation des nœuds, l'analyse de l'empreinte au sol des primitives, ou la vérification de leur validité sont autant de petites optimisations, qui, cumulées, peuvent augmenter les performances de visualisation (et donc d'interaction) de notre représentation.

3.9 Conclusions

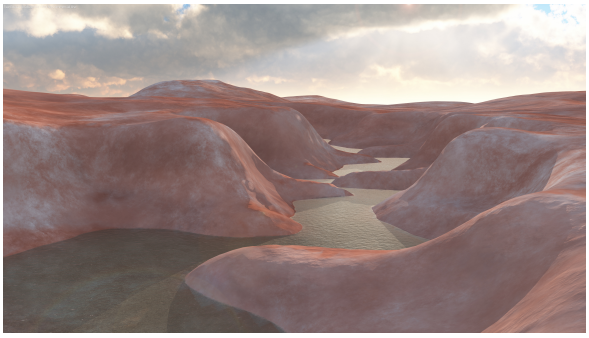


FIGURE 75 : Une scène représentant un canyon, construite avec le modèle

Avec ce modèle, nous proposons **une représentation fonctionnelle qui est contrôlable localement**. Cette structure est utilisée pour construire une fonction continue d'élévation représentant un terrain (Fig. 75). La description de la scène à l'aide d'un arbre de construction et la manipulation de modelés (unité de reliefs visuellement homogènes) rend intuitif l'édition d'une scène. Notre représentation est procédurale : elle ne stocke que peu d'informations (la structure de l'arbre, les positions des primitives, les paramètres de chaque nœud) et n'a qu'une faible empreinte mémoire.

5. Chaque arbre correspond donc à une expression algébrique.

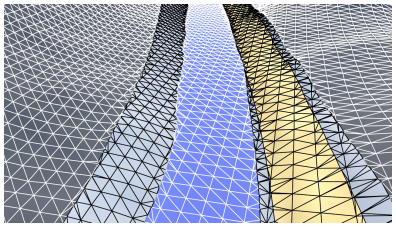
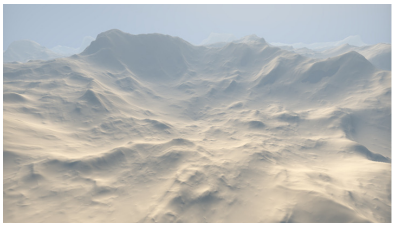
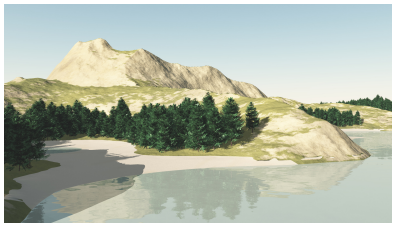
Il est possible d'étendre les fonctionnalités de ce nouveau modèle dans de nombreuses directions. Plusieurs pistes d'extensions ont déjà été explorées comme **l'intégration d'un système de niveau de détails**, **la possibilité de gérer plusieurs couches de matériaux**, ou encore **la gestion des instances** en utilisant des DAGs. D'autres perspectives sont identifiées : intégration de modèles d'érosions ontogénétiques, représentation des textures, **modélisation procédurale inverse**...

Chaque arbre de construction (ou graphe orienté acyclique) représente une fonction d'élévation continue. Celle-ci peut avoir des variations d'élévation de hautes fréquences pour représenter des détails du relief. La visualisation des contours des domaines de définition de la fonction ou des couches de matériaux est également un problème difficile. Dans le Chapitre 4, nous nous intéressons à visualiser le modèle de représentation continue à l'aide de techniques d'extraction de surfaces ou de lancer de rayons.

Dans le Chapitre 5, nous utilisons ce modèle de représentation comme support pour un algorithme de génération procédurale de terrains, structurés autour de leurs réseaux hydrographiques.

4

Visualisation

Sommaire

Vision without implementation
is hallucination.

— Auteur inconnu

Dans ce chapitre, nous cherchons à visualiser le modèle de terrain par arbres de construction (décrit Chapitre 3). Cette représentation permet de construire facilement des fonctions continues en manipulant des primitives qui ont une influence locale sur le relief et qui sont combinées à l'aide d'opérateurs intuitifs. Même si l'on a une représentation vectorielle qui définit une fonction d'élévation continue, visualiser sans artefact et de façon efficace cette fonction est un tâche complexe. Pour rendre une scène 3D, on discrétise généralement les données pour pouvoir traiter des ensembles de triangles. Le processus de discrétisation est fortement dépendant de la précision des données à afficher : si l'échantillonnage n'est pas suffisamment fin, on perd des détails. La précision du modèle n'est pas le seul critère de qualité : en effet, si le nombre de triangles projetés par pixel est trop important, on se retrouve vite avec des problèmes d'*aliasing* géométrique.

Nous proposons deux solutions principales pour le rendu de nos modèles de terrains :

- une méthode d'extraction de maillages (Section 4.1) sur CPU qui utilise l'API OpenGL pour projeter les triangles à l'écran. L'utilisation de maillages réguliers permet de visualiser rapidement la fonction de terrains. Combiné à des techniques de subdivision, il est possible de découper les frontières entre matériaux et ainsi éviter des artefacts visuels.
- une autre approche (Section 4.2), directe, par lancer de rayons, qui permet de rendre les modèles de terrains de manière quasi-exacte. La méthode proposée adapte l'algorithme du *sphere tracing* qui est une technique robuste pour visualiser des surfaces implicites dans l'espace.

Nous présentons une analyse de ces deux méthodes de rendu accompagnée de quelques résultats sous forme de temps de calcul (Section 4.3). Les limitations actuelles de ces méthodes de visualisation ainsi que plusieurs perspectives intéressantes sont abordées (Section 4.4) avant de conclure (Section 4.5).

Ces travaux ont fait l'objet de deux publications [141, 142]. Un mémoire de stage [58], qui détaille en partie l'architecture GPU du code de visualisation, complète cette section.

4.1 Maillage multi-matériaux

Un arbre de construction représente une fonction qu'il est possible d'évaluer en un point pour obtenir son altitude. Une technique pour visualiser des terrains revient à construire un maillage planaire et à élever individuellement chacun des sommets avec l'arbre. Notre système propose plusieurs outils qui construisent des maillages planaires comme des grilles régulières, des grilles adaptatives, des maillages paramétrés autour de squelettes (point, courbes) et enfin une technique plus générique de raffinement de maillages pour la gestion des multi-matériaux.

4.1.1 Maillages planaires non-dépendants de l'arbre

Il est possible d'évaluer l'arbre de construction sur l'ensemble des sommets d'un maillage quelconque pour visualiser le terrain. Certains maillages sont extrêmement rapides à construire et leur topologie n'est pas stockée explicitement, alors que d'autres sont plus complexes à gérer mais permettent d'obtenir des triangles de meilleure qualité.

La grille régulière est un des moyens les plus rapides de visualiser le terrain. Sa topologie est codée implicitement et ces grilles sont facilement exportables sous la forme d'images 2D. Cette structure est donc utilisée quand on souhaite calculer une texture à partir d'un arbre de construction. La grille peut être par la suite subdivisée si l'on souhaite obtenir les frontières exactes des différentes couches de matériaux ou du domaine de définition.

Il peut être intéressant d'utiliser une grille adaptative. Cette structure permet de représenter avec une grille plus grossière les éléments qui ne nécessitent pas forcément une grande précision. Le critère de raffinement est dépendant de la distance à la caméra. Une des structures proposées est le *quadtree* adaptatif, inspiré de Dupuy *et al.* [105] et dont le critère de subdivision est amélioré par Perrier *et al.* [293]. Cette structure repose sur un codage implicite de la coupe du *quadtree* (ce qui permet de savoir quelles sont les cellules qui sont subdivisées) sous la forme d'un code de Morton. Cette structure peut être directement calculée sur le GPU en quelques millisecondes.

4.1.2 Maillage planaire de Delaunay dépendant de l'arbre

Il est possible d'utiliser des maillages planaires quelconques pour la visualisation. Une des solutions consiste à utiliser des maillages de Delaunay qui ont la propriété d'avoir des triangles relativement équilibrés (de formes proches des triangles équilatéraux).

Pour construire un maillage de Delaunay uniforme, il faut un nuage de points qui respecte une distribution de Poisson (*i. e.* dont le spectre de puissance correspond à un bruit bleu). En pratique, un bruit bleu permet d'éviter que deux points soient trop près l'un de l'autre, ce qui empêchera la formation de triangles trop allongés.

Dans le cas d'un maillage adaptatif, on souhaite que le nombre d'échantillons soit plus important dans certaines zones (par exemple, celles dont le relief est plus escarpé). Ce processus peut être guidé par une fonction d'échantillonnage préférentiel qui est construite manuellement ou automatiquement à l'aide d'un arbre de construction.

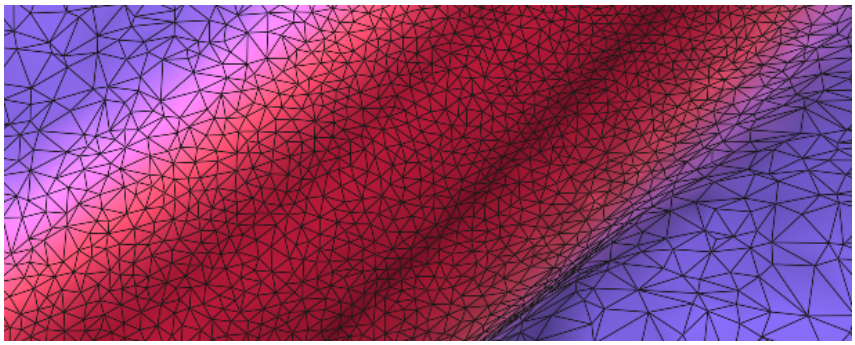


FIGURE 76 : La définition d'une fonction d'importance (elle même représentée par un arbre de construction) permet de construire des maillages adaptatifs.

Échantillonnage préférentiel

Le but de l'échantillonnage préférentiel (*importance sampling*) est de guider la fonction d'échantillonnage qui sert à placer des sommets lors de la construction d'un maillage afin d'obtenir un maximum d'échantillons dans les zones où cela est nécessaire (Fig. 76). Contrairement à ce qui se produit lors d'un échantillonnage régulier, l'échantillonnage préférentiel permet d'obtenir une plus grande densité de triangles dans les zones où le relief comporte des détails géométriques. La fonction β définissant l'importance peut être construite à l'aide d'un arbre de construction spécifique (Fig. 77) ; le calcul de l'importance peut donc s'appuyer sur l'élévation ou sur le gradient du relief. Dans notre système, l'échantillonnage préférentiel fonctionne en construisant un nuage de points respectant une distribution de Poisson maximale, puis, en élaguant cet ensemble de points avec un test dépendant de l'importance de chaque échantillon.

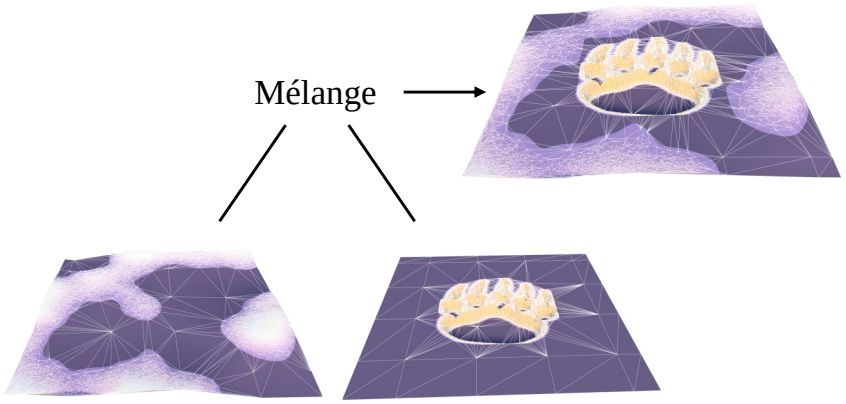


FIGURE 77 : La fonction d'importance permet de définir quelles zones doivent être maillées plus finement. Cette fonction peut être construite à l'aide d'un arbre de construction.

En pratique, on génère un nombre important de points sur la zone que l'on souhaite visualiser. Ce tirage cherche à respecter une distribution de Poisson maximale. Dans notre implémentation, nous approximons cette distribution en utilisant une technique de lancer de fléchettes (*dart throwing*) avec laquelle nous échantillonnons itérativement et uniformément la surface et où nous

invalidons les nouveaux points qui ne sont pas suffisamment éloignés des anciens échantillons. Une autre technique implantée consiste à faire une relaxation de Lloyd sur un nuage de points.

Une fois le tirage de Poisson construit, on attribue à chaque échantillon un numéro tiré aléatoirement. À partir de ce numéro et du nombre total de points de Poisson présents, nous pouvons obtenir le rang chaque échantillon, normalisé dans l'intervalle $[0;1]$. Sur l'ensemble de tous les échantillons de Poisson, nous ne gardons qu'un nombre limité d'échantillons (par exemple, 70% des points). Cela correspond à construire l'ensemble des échantillons valides :

$$\left\{ \mathbf{p}_i \quad | \quad \frac{i}{n} \leq \beta \right\}$$

Si $\beta = 1$, l'ensemble des points est conservé et si $\beta = 0$, seul le premier point d'indice 0 l'est.

Échantillonnage contraint

Certaines primitives géométriques ont besoin d'être échantillonnées précisément (par exemple, les lignes de crêtes ou les bords d'une rivière). Comme la probabilité de tirer un point sur une primitive curviligne est nulle, il est nécessaire de passer par des mécanismes plus déterministes (Fig. 78).

Chaque primitive peut avoir une fonction d'échantillonnage contrainte qui permet de générer un ensemble de n points paramétrés. Une primitive courbe peut par exemple être échantillonnée uniformément selon l'abscisse curviligne.

FIGURE 78 : L'utilisation d'échantillons contraints permet de préserver les éléments saillants (*sharp features*) et d'éviter de nombreux artefacts visuels apparaissant aux frontières entre plusieurs couches de matériaux.

Dans notre système, nous triangulons l'union de tous ces échantillons contraints avec les points de Poisson déjà construits. Il est important de noter que ce nouveau nuage de points ne respecte plus une distribution de Poisson maximale et peut aboutir à la création de triangles de Delaunay allongés. Un algorithme plus efficace et plus flexible permettant de mêler échantillonnage de Poisson contraint et échantillonnage adaptatif est présenté par Corsini *et al.* [81].

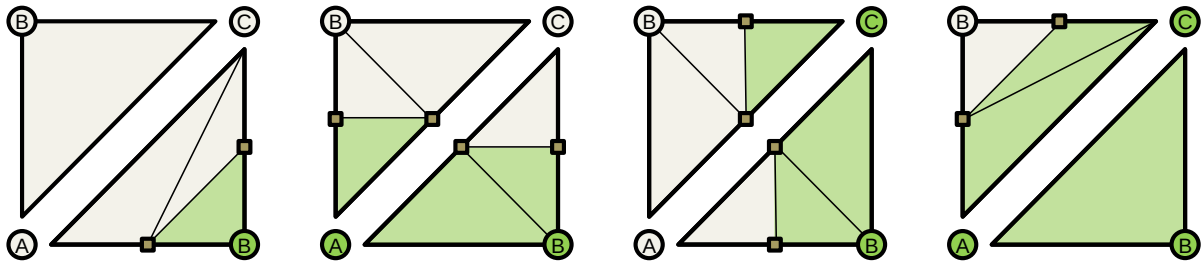


FIGURE 79 : Les triangles sont découpés selon 8 motifs distincts pour trouver la frontière d'une fonction continue (en vert) dont on connaît la valeur aux sommets. Les points marron correspondent à la frontière (définie par une isovaleur) ; leur position est calculée par une évaluation dichotomique de la fonction à découper le long de l'arête.

4.1.3 Raffinement de maillage

Une autre technique de visualisation consiste à construire un maillage puis de le raffiner selon un critère de subdivision ou pour échantillonner une frontière curviligne.

Découpe de frontières

Les terrains que nous créons ont un support compact (correspondant au domaine Ω_T) : ce support ne colle pas forcément aux bords du maillages que nous évaluons avec l'arbre de construction. Pour remédier à ce problème, il est possible de découper le maillage pour trouver le bord du domaine et ainsi éliminer les triangles se trouvant en dehors du domaine.

La découpe de frontières intervient lorsqu'on est en présence de plusieurs couches de matériaux différents. Ces couches de matériaux ont un bord (la frontière où leur épaisseur devient nulle). Il est nécessaire de trianguler ces bords pour éliminer des artefacts visuels. Ces artefacts sont particulièrement visibles sur des grilles régulières. Une couche de matériaux définie sur une diagonale peut sembler visuellement crénelée. Des problèmes plus complexes peuvent apparaître, par exemple, quand on souhaite représenter les bords d'une rivière. Il y a, en effet, très peu de chances que les points du maillage que l'on souhaite élever soient parfaitement placés sur les berges de la rivière (sans utiliser un échantillonnage contraint). Ainsi, il est nécessaire de raffiner ces triangles pour que le maillage d'eau soit plan et intersecte la berge au bon endroit.

Le *marching square* s'intéresse à subdiviser des grilles régulières (par exemple des images en niveaux de gris) pour obtenir des isocontours valides. Dans notre cas, nous adaptons cet algorithme pour découper des triangles (pour pouvoir raffiner n'importe quel maillage). En pratique, dans le cas de la fonction de poids et d'une découpe de Ω_T, on cherche les triangles possédant à la fois des sommets dont le poids est inférieur à la valeur T et des sommets dont le poids est supérieur à cette valeur (Fig. 79). La frontière du domaine intersecte alors au moins deux arêtes de ce triangle.

Pour trouver le point de passage exact, deux techniques sont possibles : on peut considérer les valeurs d'épaisseurs des différentes couches de matériaux au niveau des sommets et interpoler linéairement chaque couche. Cette méthode est très rapide, mais on se base sur une hypothèse

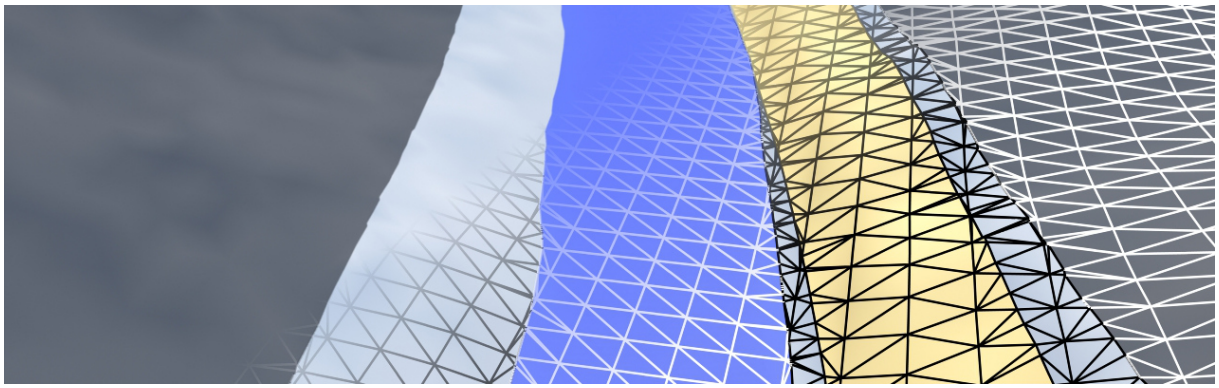


FIGURE 80 : La découpe de maillages permet d'obtenir des frontières propres entre les matériaux. Elle est également utilisée pour découper le bord du domaine Ω_T visible sur la Fig. 71.

presque toujours fausse : la fonction dont on cherche un isocontour varierait linéairement, or ce n'est presque jamais le cas. Dans le cas de fluides, cette interpolation linéaire peut provoquer des artefacts visuels particuliers.

Une autre technique, plus lente mais beaucoup plus précise, consiste à évaluer la fonction représentée par un arbre de construction sur une série de points le long de l'arête, pour trouver la valeur-seuil correspondant à la frontière. Pour accélérer ce processus, il est possible de procéder par une recherche dichotomique.

Si le raffinement du maillage nous permet de retrouver un point correspondant au seuil recherché, c'est-à-dire que nous arrivons à placer un point sur la frontière (Fig. 80), il est possible que la fonction que l'on souhaite découper présente des variations de plus hautes fréquences. En effet, la fonction dont on souhaite retrouver une isovaleur peut osciller plusieurs fois autour de cette isovaleur. Même si notre algorithme n'est pas capable de capturer ces multiples frontières présentes le long du segment, le résultat visuel produit ne présentera pas d'artefacts visuels. Le corollaire est que toute fonction dont on souhaite retrouver la frontière est impossible à échantillonner si son domaine ne comprend aucun des sommets initiaux du maillage à découper (*i. e.* le domaine de la fonction est par exemple contenu dans un triangle).

Subdivisions

En plus de trouver les frontières des fonctions représentant les différentes couches de matériaux, il peut être nécessaire, pour obtenir une visualisation correcte, de subdiviser (ou simplifier) le maillage selon des critères géométriques (Fig. 81).

Dans notre système, les critères de subdivisions peuvent être portés par des arêtes (*e. g.* un test sur la taille de l'arête, ou bien sur la différence d'élévation) ou par les triangles (*e. g.* un test sur leur aire ou sur leur orientation par rapport à la caméra). Il est également possible de faire des tests plus compliqués qui peuvent faire intervenir des calculs de simulation, ou la position de la caméra ou d'un agent par exemple. Avec une structure de maillage plus complexe, comme des cartes combinatoires, il serait aussi envisageable de faire des insertions de sommets.

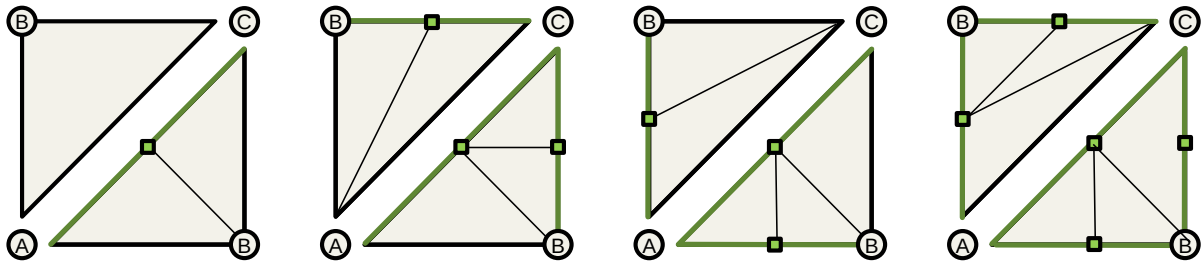


FIGURE 81 : Les triangles peuvent également être subdivisés avec un critère porté par les arêtes (longueur trop grande, différence d'altitude trop grande, ...). Les nouveaux points insérés peuvent être positionnés au milieu de l'arête ou par une procédure dichotomique (par exemple pour créer deux segments avec le même différentiel en altitude).

4.2 Lancer de rayons

Dans cette partie, nous présentons une méthode de lancer de rayons pour visualiser le modèle de terrain décrit dans le Chapitre 3 en tirant parti des propriétés de construction par arbre de fonctions. Notre approche s'inspire de la méthode de *sphere tracing* [155] qui est une technique robuste pour visualiser des surfaces implicites dans l'espace à l'aide d'un lancer de rayons. Contrairement aux LG-surfaces [180], il n'est pas nécessaire de disposer d'une fonction de classe C^2 : il suffit d'avoir une fonction-potentiel de classe C^0 qui respecte la propriété de Lipschitz. La dérivée n'est pas nécessairement continue, ni même définie sur tout le domaine.

Rappelons qu'une fonction dans l'espace $h : \mathbf{R}^3 \to \mathbf{R}$ garantit la propriété de Lipschitz sur le domaine Ω, si, et seulement si, il existe une constante positive λ telle que :

$$\forall (\mathbf{p}, \mathbf{q}) \in \Omega \times \Omega, \ |h(\mathbf{p}) - h(\mathbf{q})| < \lambda \|\mathbf{p} - \mathbf{q}\|.$$

La constante de Lipschitz λ est la valeur minimale satisfaisant cette équation. Elle correspond au coefficient de la pente maximale que peut atteindre la fonction h. En pratique, les constantes de Lipschitz sont sur-estimées par une borne de Lipschitz, en particulier pour les opérateurs quand les bornes de leurs sous-arbres sont connues.

4.2.1 Algorithmes classiques

Dans cette section, nous présentons les principes du lancer de rayons et des quelques algorithmes utilisés. [1]

Il n'est pas toujours possible de trouver une solution exacte (*i. e.* de manière analytique) au calcul d'intersection entre un rayon et un objet quelconque. Si le lancer de rayons classique

1. Il est important de ne pas confondre des techniques de rendu avec des méthodes pour calculer l'éclairage [417] comme le *path tracing*, le *bi-directional path tracing*, le *Metropolis light transport*, ou encore le *photon mapping*. La plupart de ces algorithmes de calcul de l'éclairage se basent sur la notion de lancer de rayons classique (*ray casting*, ou quand il y a des rebonds, *ray tracing*) pour calculer le chemin de la lumière.

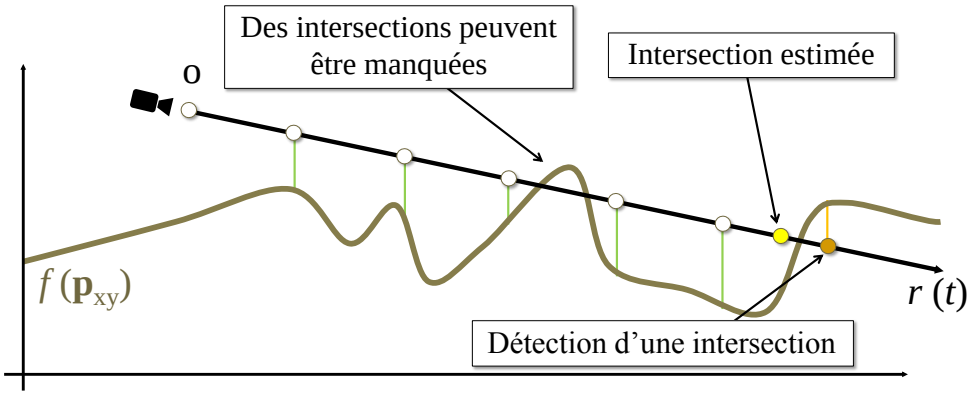


FIGURE 82 : Le *ray marchinbg* est utilisé quand il n'est pas possible de calculer une solution analytique car la primitive visualisée est trop complexe. Avec cet algorithme, on avance le long de chaque rayon avec un pas constant. Il est possible de rater certaines intersections avec des objets de la scène.

fonctionne très bien avec des triangles (et par extension, des objets maillés), certaines primitives implicites complexes (comme les terrains générés par des fonctions de bruits) nécessitent des méthodes alternatives comme le *ray marching* ou le *sphere tracing* pour calculer les intersections avec les rayons. De la même manière, des algorithmes spécialisés pour le rendu de terrains existent [137, 160, 313, 377] et on retiendra particulièrement le *Grid tracing* [252] et le *QAEB tracing* [106].

4.2.1.1 *Ray marching*

Un terrain est défini généralement par une fonction de bruit complexe. Il est difficile, voire impossible, de calculer une intersection entre l'équation du rayon et la fonction définissant l'altitude du terrain. On utilise donc d'autres mécanismes pour pouvoir visualiser le relief. Le plus simple est celui du *ray marching* (Fig. 82) qui consiste à avancer le long du rayon avec un incrément (autrement dit, un pas).

Ce pas, s'il est suffisamment petit, permet d'obtenir une image relativement fidèle au terrain au détriment des performances. À l'inverse, si le pas est très grand, il accélère le temps de rendu de l'image mais peut introduire des artefacts visuels. En effet, cet incrément peut faire manquer une intersection à l'algorithme.

Le *ray marching* permet également de visualiser des effets volumiques : la luminosité émise par un point peut être atténuée le long du rayon quand on traverse un nuage ou un liquide. Plus le pas sera fin, plus le rendu volumique sera de bonne qualité.

4.2.1.2 *Sphere tracing*

Le *sphere tracing* consiste à marcher le long d'un rayon Δ, depuis une caméra $\mathbf{o}$ vers la surface, avec un incrément adaptatif $|h(\mathbf{p})| / \lambda$ suffisamment petit pour ne jamais traverser la surface

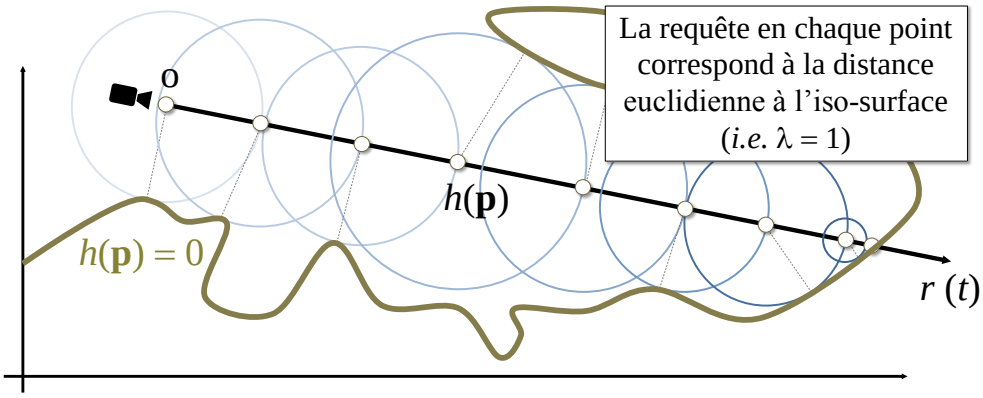


FIGURE 83 : Le *sphere tracing* permet d'avancer le long de chaque rayon avec un pas qui varie de manière à garantir qu'on ne rate aucune intersection avec le terrain. À l'inverse, quand l'environnement est vide, le pas devient grand et on avance très vite. Le *sphere tracing* fonctionne à partir du moment où la fonction que l'on souhaite visualiser est Lipschitzienne (c'est-à-dire dont on peut borner la dérivée). Quand $\lambda = 1$, l'algorithme peut se visualiser avec des sphères.

(Fig. 83). On utilise une valeur seuil ϵ à partir de laquelle on décide qu'on est suffisamment proche de la surface pour considérer une intersection entre le rayon et le terrain. Soit $r(t) = \mathbf{o} + t\,\mathbf{u}$ l'équation paramétrique du rayon (où $\mathbf{u}$ est normalisé). Nous notons $h_\Delta(t) = h \circ r(t)$ la fonction-potentiel le long du rayon.

L'algorithme du *sphere tracing* consiste à :

1. Calculer une borne de Lipschitz λ de la fonction h, et l'intervalle $[t_-, t_+] = \Delta \cap \Omega$. Initialiser $t = \max(0, t_-)$.
2. Tant que $t < t_+$, calculer $h_\Delta(t)$.
   - 2.1 Si $|h_\Delta(t)| < \varepsilon$, alors nous considérons qu'il y a intersection.
   - 2.2 Sinon progresser de l'incrément t avec $|h_\Delta(t)| / \lambda$ et continuer.

Cet algorithme permet de rendre des surfaces implicites. La plupart du temps, les scènes comportant des surfaces implicites sont simples et définissent les primitives génératrices à l'aide d'une distance euclidienne. Ainsi, la distance varie de manière linéaire, son gradient est borné sur l'intégralité de la scène et la constante de Lipschitz est exactement $\lambda = 1$ [2]. Dans le cas d'un terrain, la constante de Lipschitz décrit un cône (Fig. 84).

Il existe des optimisations pour l'algorithme du *sphere tracing* qui permettent d'avancer, en moyenne, plus vite le long de chaque rayon [184]. Il est également possible de s'appuyer sur d'autres stratégies comme le *cone stepping* [103, 303] ou le *cone marching* [364] pour accélérer les calculs.

2. Ce cas particulier est très souvent utilisé dans les scènes réalisées par les *demomakers*.

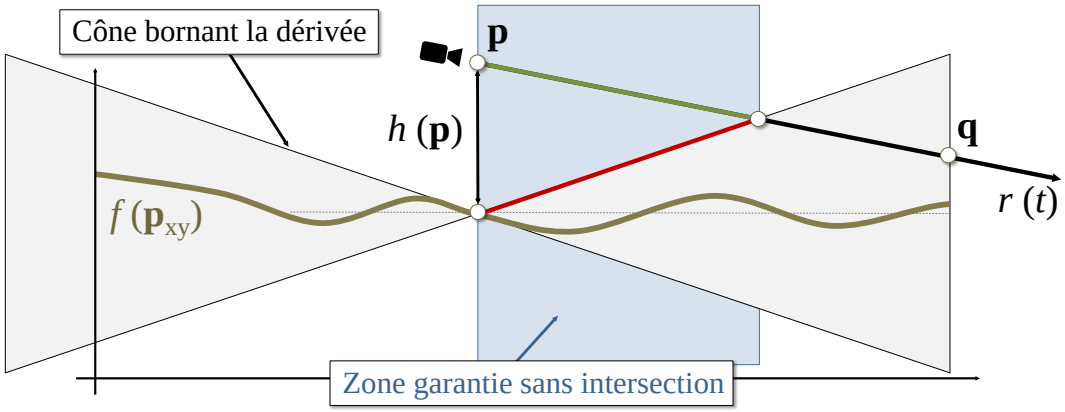


FIGURE 84 : Le calcul de la constante de Lipschitz λ permet de borner la variation du relief défini par la fonction f. Cela permet de garantir que dans le cas le plus extrême, le terrain sera tangent au segment rouge. Ainsi, on peut définir un pas le long de la direction du rayon $\mathbf{u}$ (représenté par le segment vert) qui garantit qu'on ne ratera aucune intersection avec le terrain.

4.2.2 Extension du *sphere tracing*

Soit $\mathbf{p}$ un point de $\mathbf{R}^3$, on note $\mathbf{p}_{xy}$ et $\mathbf{p}_z$ les coordonnées x, y et z de $\mathbf{p}$; $f(\mathbf{p}_{xy})$ représentant l'élévation du terrain en $\mathbf{p}_{xy}$. Nous définissons la surface implicite représentant notre terrain à l'aide de la fonction potentiel $h : \mathbf{R}^3 \to \mathbf{R}$:

$$h(\mathbf{p}) = \mathbf{p}_z - f(\mathbf{p}_{xy}).$$

La fonction f étant construite par une combinaison hiérarchique de primitives définies sur des supports compacts, on peut chercher des constantes de Lipschitz locales. Ces dernières seront en général très inférieures à la constante de Lipschitz globale λ. Des bornes plus petites et donc plus efficaces peuvent être calculées en utilisant des structures de décomposition de l'espace au prix d'un surcoût-mémoire élevé. Nous proposons un algorithme de *sphere tracing* adapté s'appuyant sur une évaluation à la volée d'une borne de Lipschitz locale en tout point du rayon.

L'algorithme consiste à avancer progressivement et de manière adaptative le long du rayon en calculant une borne de Lipschitz locale (valide au voisinage du point considéré). À chaque étape, nous essayons d'avancer d'un pas δ et calculons la distance effective que nous allons parcourir $s(t,\delta)$. Cette distance définit un intervalle $[t, t+\delta]$ correspondant à un segment $[\mathbf{p}, \mathbf{q}]$ où $\mathbf{p}$ correspond au point actuel sur le rayon, et où $\mathbf{q} = \mathbf{p} + \delta\,\mathbf{u}$ est un point placé plus loin sur le rayon (Fig. 84).

Nous effectuons une requête dans l'arbre pour calculer à la fois $f(\mathbf{p})$ et la borne de Lipschitz locale $\lambda(t,\delta)$ sur l'intervalle $[t, t+\delta]$. La fonction $h_\Delta(t)$ est développée ainsi :

$$h_\Delta(t) = \mathbf{o}_z + t\,\mathbf{u}_z - f(\mathbf{o}_{xy} + t\,\mathbf{u}_{xy}).$$

La dérivée de $h_\Delta(t)$ par rapport à t est :

$$h_\Delta{}'(t) = \mathbf{u}_z - ||\,\mathbf{u}_{xy}\,||\;\; f'(\mathbf{o}_{xy} + t\,\mathbf{u}_{xy}).$$

En fonction de la valeur de $\mathbf{u}_z - ||\,\mathbf{u}_{xy}\,||\,\lambda(t,\delta)$, deux cas peuvent se produire :

1. Si $\mathbf{u}_z - ||\,\mathbf{u}_{xy}\,||\,\lambda(t,\delta) \geq 0$, la dérivée de la fonction est positive, ce qui veut dire que la distance entre le rayon et le terrain augmente. Par conséquent, aucune intersection ne peut se produire sur l'intervalle $[t, t+\delta]$.
$$s(t,\delta) = \delta\,.$$
2. Si $\lambda(t,\delta)\,||\,\mathbf{u}_{xy}\,|| - \mathbf{u}_z \leq 0$, une zone garantie sans intersection est détectée et il est possible d'avancer de la distance :
$$s(t,\delta) = \frac{h_\Delta(t)}{|\,\mathbf{u}_z - ||\,\mathbf{u}_{xy}\,||\,\lambda(t,\delta)\,|}\,.$$

Par conséquent, nous pouvons avancer le long du rayon sans traverser la surface du terrain en progressant de la distance minimale suivante :

$$s(t,\delta) = \min\left(\delta, \frac{h_\Delta(t)}{|\,\mathbf{u}_z - ||\,\mathbf{u}_{xy}\,||\,\lambda(t,\delta)\,|}\right).$$

Étant donnée une distance de progression initiale δ, l'algorithme procède ainsi :

1. Calculer l'intervalle entre le rayon et le domaine $[t_-, t_+] = \Delta \cap \Omega$. Initialiser $t = \max(0, t_-)$.
2. Tant que $t < t_+$, calculer $h_\Delta(t)$.
   2.1 Si $h_\Delta(t) < \epsilon$ alors, on considère qu'il y a une intersection en t.
   2.2 Sinon, trouver $\lambda(t,\delta)$ et utiliser la borne pour augmenter t avec l'incrément $s(t,\delta)$.
   2.3 Mettre à jour la distance δ avec 2δ.

L'étape 2.3 de l'algorithme contraint à essayer d'avancer constamment avec un pas de plus en plus grand. Le calcul de $s(t,\delta)$ garantit que ce pas est ajusté de manière à ne jamais traverser la surface.

4.2.3 Calcul des constantes de Lipschitz

Notre modèle permet d'obtenir une évaluation des constantes de Lipschitz pour tous les types de nœuds internes (opérateurs) et de feuilles (primitives) de l'arbre. Par conséquent, la constante de Lipschitz λ de la racine de l'arbre est calculée en traversant la structure de l'arbre de manière récursive. Pour chaque nœud nous avons besoin de calculer les élévations maximum et minimum et les constantes de Lipschitz des fonctions d'élévation et de poids.

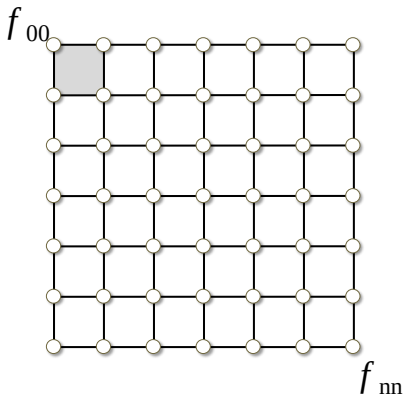
4.2.3.1 Primitives

Pour chaque type de primitive, il est possible de définir une constante de Lipschitz de manière analytique. Son calcul dépend des paramètres qui interviennent dans la fonction d'élévation (les paramètres de bruits, d'inclinaison, *etc*). La constante qui nous intéresse le plus concerne la borne λ de la dérivée de la fonction d'élévation f. Il est malgré tout nécessaire de calculer une borne λ_α de la dérivée de la fonction de poids α (qui est utilisée pour le calcul des bornes λ de certains opérateurs).

Toutes les primitives ont de bonnes propriétés de continuité et de dérivabilité et respectent toujours la propriété de Lipschitz.

Constante de Lipschitz d'une carte de hauteurs

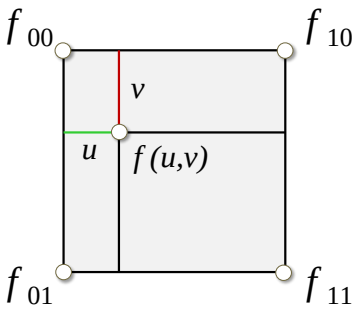
Pour les primitives construites à partir d'images, la fonction d'élévation est calculée par une interpolation bilinéaire ou par une interpolation bicubique contrainte des données contenues dans la carte de hauteurs. Considérons le calcul de la constante de Lipschitz dans le cas de la projection de la carte de hauteurs sur un domaine carré. Lors d'une projection le long d'une courbe ou sur un quadrangle quelconque, cette constante est modifiée par la déformation.



Considérons une carte de hauteurs de taille $n \times n$.

Interpolation bilinéaire

Étudions le cas d'une interpolation bilinéaire de l'image. L'interpolation bilinéaire comme l'interpolation bicubique contrainte (dont les dérivées première et seconde sont nulles en chaque point) ne nécessitent pas d'examiner les valeurs voisines du carreau traité. On en revient donc à l'étude d'un quadrangle dont on connaît les valeurs aux 4 sommets. On rappelle que l'interpolation bilinéaire d'un carreau ne produit pas un plan mais bien une surface non linéaire (*i. e.* une parabole le long de la diagonale).



Quand elle est réalisée dans un carré-unité, une interpolation bilinéaire peut s'écrire sur le domaine $(u,v) \in [0,1]^2 = \Omega$ sous la forme :

$$\begin{aligned} f(u,v) &= f_{00}\,(1-u)\,(1-v) \;+\; f_{10}\,u\,(1-v) \;+\; f_{01}\,(1-u)\,v \;+\; f_{11}\,u\,v \\ &= f_{00} \;+\; (f_{10}-f_{00})\,u \;+\; (f_{01}-f_{00})\,v \;+\; (f_{00}-f_{10}-f_{01}+f_{11})\,u\,v \end{aligned}$$

Le gradient s'écrit :

$$\nabla f(u,v) = \begin{pmatrix} (f_{10}-f_{00}) + (f_{00}-f_{10}-f_{01}+f_{11})\,v \\ (f_{01}-f_{00}) + (f_{00}-f_{10}-f_{01}+f_{11})\,u \end{pmatrix}$$

On cherche à borner la norme du gradient notée $||\nabla f(u,v)||$ sur Ω. On remarque que $||\nabla f(u,v)||^2$ est une fonction positive convexe, donc le maximum est atteint sur le bord du domaine $\partial\Omega$.

De plus $||\nabla f||^2$ définit un paraboloïde dont l'intersection avec chacun des segments du bord du domaine définit une parabole. Donc le maximum est atteint sur l'un des 4 points $(0,0)$, $(0,1)$, $(1,0)$ ou $(1,1)$ du domaine Ω. On recherche par conséquent :

$$\lambda^2 = \max_{(u,v)\in\{(0,0),(0,1),(1,0),(1,1)\}} ||\nabla f(u,v)||^2$$

$$||\nabla f(u,v)||^2 = \begin{array}{r} \left((f_{10}-f_{00})+(f_{00}-f_{10}-f_{01}+f_{11})\,v\right)^2 \\ +\left((f_{01}-f_{00})+(f_{00}-f_{10}-f_{01}+f_{11})\,u\right)^2 \end{array}$$

Les 4 cas qui se présentent aux coins se simplifient [3] en :

$$\begin{aligned}
{\lambda_{00}}^2 &= \left((f_{10}-f_{00})+\cancel{(f_{00}-f_{10}-f_{01}+f_{11})\times 0}\right)^2+\left((f_{01}-f_{00})+\cancel{(f_{00}-f_{10}-f_{01}+f_{11})\times 0}\right)^2 \\
&= (f_{10}-f_{00})^2+(f_{00}-f_{01})^2 \\
{\lambda_{10}}^2 &= \left((f_{10}-f_{00})+\cancel{(f_{00}-f_{10}-f_{01}+f_{11})\times 0}\right)^2+\left((\cancel{f_{01}}-\cancel{f_{00}})+(\cancel{f_{00}}-f_{10}-\cancel{f_{01}}+f_{11})\times 1\right)^2 \\
&= (f_{10}-f_{00})^2+(f_{11}-f_{10})^2 \\
{\lambda_{01}}^2 &= \left((\cancel{f_{10}}-\cancel{f_{00}})+(\cancel{f_{00}}-\cancel{f_{10}}-f_{01}+f_{11})\times 1\right)^2+\left((f_{01}-f_{00})+\cancel{(f_{00}-f_{10}-f_{01}+f_{11})\times 0}\right)^2 \\
&= (f_{01}-f_{11})^2+(f_{00}-f_{01})^2 \\
{\lambda_{11}}^2 &= \left((\cancel{f_{10}}-\cancel{f_{00}})+(\cancel{f_{00}}-\cancel{f_{10}}-f_{01}+f_{11})\times 1\right)^2+\left((\cancel{f_{01}}-\cancel{f_{00}})+(\cancel{f_{00}}-f_{10}-\cancel{f_{01}}+f_{11})\times 1\right)^2 \\
&= (f_{01}-f_{11})^2+(f_{11}-f_{10})^2
\end{aligned}$$

Nous obtenons :

$$\lambda^2 = \max({\lambda_{00}}^2,{\lambda_{10}}^2,{\lambda_{01}}^2,{\lambda_{11}}^2)$$

Soit δ^2 le maximum des différences au carré entre les élévations des points adjacents :

$$\delta^2 = \max\left((f_{10}-f_{00})^2,(f_{11}-f_{10})^2,(f_{01}-f_{11})^2,(f_{00}-f_{01})^2\right)$$

On peut maintenant écrire $\lambda^2 \leq 2\,\delta^2$ d'où $\lambda \leq \sqrt{2}\,\delta$ avec

$$\delta = \max\left(|f_{10}-f_{00}|,|f_{11}-f_{10}|,|f_{11}-f_{01}|,|f_{11}-f_{10}|\right)$$

Bien que $\sqrt{2}\,\delta$ corresponde à une borne supérieure de la constante de Lipschitz λ, cette valeur est plus rapide à calculer.

Le calcul de cette borne simplifiée peut être optimisé en regroupant sur toute l'image l'ensemble des arêtes verticales et l'ensemble des arêtes horizontales :

$$\lambda \leq \sqrt{2}\max_{i,j\in[0,n-1]}\left(|f_{i,j}-f_{i+1,j}|,|f_{i,j}-f_{i,j+1}|\right).$$

On a donc un algorithme simple pour calculer la constante de Lipschitz pour une fonction construite à partir de l'interpolation bilinéaire d'une carte de hauteurs.

Constante de Lipschitz d'une fonction de bruit

Les fonctions de bruits sont généralement des fonctions C^1. Les bruits par valeurs sont intrinsèquement construits en interpolant des valeurs discrètes. En pratique, la constante de Lipschitz va varier selon la fonction d'interpolation, la distance entre deux échantillons et l'amplitude maximale des valeurs aléatoires.

3. On réordonne les termes grâce à : $(a-b)^2=(b-a)^2$

Constante de Lipschitz d'une somme de bruits

Dans notre modèle, les primitives à squelettes sont souvent construites par combinaison de sommes de fonctions de bruit à différentes amplitudes et fréquences. La fonction de bruit de gradient, notée n, respecte intrinsèquement la propriété de Lipschitz, et nous notons λ_n sa borne. Quand plusieurs harmoniques de bruits sont sommées comme dans le cas d'une primitive-disque, nous estimons la nouvelle borne par la somme des constantes de Lipschitz de chacune des harmoniques.

$$\lambda = \sum_{i=1}^{n} \frac{a_i \, \lambda_n}{s_i} .$$

Les primitives à niveaux de détails sont similaires aux primitives à squelettes mais elles filtrent les harmoniques qui décrivent les détails. Moins de détails signifie des bornes de Lipschitz plus petites et donc des temps de calculs plus rapides.

Le bruit *ridged-multifractal* n'est pas forcément de classe de continuité C^1 mais sa borne de Lipschitz est égale à celle d'une somme de bruits classique car l'introduction d'une valeur absolue dans la formule ne change pas la borne de Lipschitz.

Fonction de poids

Les fonctions de poids α sont définies par $\alpha(\mathbf{p}) = g \circ d(\mathbf{p})$ où d est la distance euclidienne à un squelette et g la fonction de Wyvill. La constante de Lipschitz de α est le maximum de la dérivée de g car d a une borne de Lipschitz égale à 1 (la norme du gradient de d est d'ailleurs égale à 1 partout, sauf sur le squelette où le gradient n'est pas défini).

Fonction de mélange de Wyvill.

Cette fonction est utilisée pour définir les poids $\alpha(\mathbf{p})$ de certaines primitives à squelettes. La fonction de mélange de Wyvill $g_2(x)$ est de classe C^2 et est définie ainsi [415] :

$$g_2(x) = \begin{cases} \left(1 - \left(\frac{x}{r}\right)^2\right)^3 & \text{si } x < r, \\ 0 & \text{sinon}. \end{cases}$$

Pour $x < r$, nous avons :

$$g_2'(x) = \frac{-6\,x\,(r^2 - x^2)^2}{r^6} .$$

Trouver la constante de Lipschitz de g consiste à trouver un extremum de g_2'. Pour cela, nous allons étudier la dérivée seconde :

$$g_2''(x) = \frac{-6\,(r^2 - x^2)\,(r^2 - 5x^2)}{r^6} .$$

La dérivée seconde a deux racines positives en r et $r/\sqrt{5}$. Seule la dernière correspond à un maximum de $g_2'(x)$. La valeur maximum de $g_2'(x)$ est alors obtenue en évaluant $g_2'(r/\sqrt{5})$, ce qui donne $\lambda = 96\sqrt{5}\,/\,125\,r$.

4.2.3.2 Opérateurs

Pour chaque type d'opérateur, nous pouvons calculer sa borne de Lipschitz en fonction des bornes des sous-arbres à combiner λ_A et λ_B. Par exemple, la constante de Lipschitz λ d'un nœud de mélange est définie comme suit :

$$\lambda \leq \max(\lambda_{f_A}, \lambda_{f_B}) + \max(\lambda_{\alpha_A}, \lambda_{\alpha_B}) \sup|f_A - f_B|,$$

où λ_{f_A}, λ_{f_B}, λ_{α_A}, λ_{α_B} représentent les constantes de Lipschitz des fonctions d'élévation et des fonctions de poids de chacun des sous-arbres respectivement.

Opérateur de mélange.

Soit A et B les deux sous-nœuds de l'opérateur de mélange. Soit $\beta = \alpha_A / (\alpha_A + \alpha_B)$. La fonction de mélange peut maintenant s'écrire comme suit :

$$f = \beta f_A + (1-\beta) f_B, \qquad \alpha = \alpha_A + \alpha_B.$$

Le domaine de définition correspond à $\Omega_0 = \{\mathbf{p} \in \mathbf{R}^2 \mid \alpha(\mathbf{p}) > 0\}$.

Soit λ_A et λ_B les constantes de Lipschitz de f_A et f_B. Nous considérons des fonctions f_A, f_B, α_A et α_B de classe C^1. La constante de Lipschitz λ de f est définie par $\lambda = \sup_{\mathbf{p}\in\Omega} ||\nabla f(\mathbf{p})||$. Le gradient ∇f peut s'écrire comme suit :

$$\begin{aligned}
\nabla f &= \nabla(\beta f_A) + \nabla((1-\beta) f_B) \\
&= \nabla(\beta f_A) + \nabla(f_B - f_B \beta) \\
&= \nabla(\beta f_A) + \nabla(f_B) - \nabla(f_B \beta) \\
&= \beta\, \nabla f_A + f_A\, \nabla\beta + \nabla f_B - \beta\, \nabla f_B - f_B\, \nabla\beta \\
&= (f_A - f_B)\, \nabla\beta + (1-\beta)\, \nabla f_B + \beta\, \nabla f_A.
\end{aligned}$$

On remarque que $(1-\beta)\, \nabla f_B + \beta\, \nabla f_A$ correspond à une interpolation : comme $\beta \in [0,1]$, on peut en déduire que l'ensemble de cette sous-expression est comprise entre ∇f_B et ∇f_A, ce qui revient à la borner par $\max(\lambda_A, \lambda_B)$. Nous avons donc :

$$||\nabla f|| \leq \sup|f_A - f_B|\, \lambda_\beta + \max(\lambda_A, \lambda_B)$$

Il nous reste à trouver la constante de Lipschitz λ_β de β. Nous avons :

$$\nabla\beta = \frac{1}{\alpha_A + \alpha_B}\Big((1-\beta)\, \nabla\alpha_A - \beta\, \nabla\alpha_B\Big).$$

Comme $(1-\beta)\, \nabla\alpha_A - \beta\, \nabla\alpha_B$ correspond à une interpolation avec $\beta \in [0,1]$, on peut en déduire que l'ensemble de cette sous-expression est comprise entre $\nabla\alpha_A$ et $\nabla\alpha_B$ ce qui revient à la borner par $\max(\lambda_{\alpha_A}, \lambda_{\alpha_B})$. Nous avons calculé λ_β et nous pouvons par conséquent écrire :

$$\lambda \leq \sup|f_A - f_B|\, \max(\lambda_{\alpha_A}, \lambda_{\alpha_B}) + \max(\lambda_A, \lambda_B).$$

La borne λ_α de la dérivée de la fonction de poids est obtenue directement :

$$\lambda_\alpha = \lambda_{\alpha_A} + \lambda_{\alpha_B}$$

Opérateur de remplacement.

Soit A et B les deux sous-nœuds de l'opérateur de remplacement qui peut s'écrire :

$$f = (1 - \alpha_B) f_A + \alpha_B f_B, \qquad \alpha = \alpha_A .$$

Le domaine de définition correspond à $\Omega_0 = \{ \mathbf{p} \in \mathbf{R}^2 \mid \alpha_A(\mathbf{p}) > 0 \}$.

Soit λ_A et λ_B les constantes de Lipschitz de f_A et f_B. La constante de Lipschitz λ de f est définie par $\lambda = \sup_{\mathbf{p}\in\Omega} ||\nabla f(\mathbf{p})||$. Le gradient ∇f peut s'écrire comme suit :

$$\begin{aligned}
\nabla f &= \nabla((1 - \alpha_B) f_A + \alpha_B f_B) \\
&= \nabla((1 - \alpha_B) f_A) + \nabla(\alpha_B f_B) \\
&= \nabla(f_A - f_A \alpha_B) + \nabla(\alpha_B f_B) \\
&= \nabla f_A - \nabla(f_A \alpha_B) + f_B \nabla \alpha_B + \alpha_B \nabla f_B \\
&= \nabla f_A - \alpha_B \nabla f_A - f_A \nabla \alpha_B + f_B \nabla \alpha_B + \alpha_B \nabla f_B \\
&= (1 - \alpha_B) \nabla f_A + \alpha_B \nabla f_B + (f_B - f_A) \nabla \alpha_B .
\end{aligned}$$

L'on peut remarquer que $(1 - \alpha_B) \nabla f_A + \alpha_B \nabla f_B$ correspond à une interpolation : comme $\alpha_B \in [0, 1]$ (la valeur peut être ramenée dans l'intervalle sans changer la signification de l'opérateur) on peut en déduire que l'ensemble de cette sous-expression est comprise entre ∇f_A et ∇f_B ce qui revient à la borner par $\max(\lambda_A, \lambda_B)$. Nous avons donc :

$$||\nabla f|| \leq \max(\lambda_A, \lambda_B) + \sup|f_B - f_A| \lambda_{\alpha_B}$$

Il est également nécessaire de calculer un borne de Lipschitz pour la fonction de poids. Quand $\alpha = \alpha_A$, le résultat est trivial, mais dans le cas où $\alpha = (1 - \alpha_B)\alpha_A + {\alpha_B}^2$, il est nécessaire de calculer une nouvelle constante. Le gradient $\nabla\alpha$ peut s'écrire :

$$\begin{aligned}
\nabla\alpha &= \nabla((1 - \alpha_B)\alpha_A + {\alpha_B}^2) \\
&= \nabla(\alpha_A - \alpha_A \alpha_B) + \nabla({\alpha_B}^2) \\
&= \nabla\alpha_A - \nabla(\alpha_A \alpha_B) + \nabla({\alpha_B}^2) \\
&= \nabla\alpha_A - \alpha_A \nabla\alpha_B - \alpha_B \nabla\alpha_A + 2\alpha_B \nabla\alpha_B \\
&= (1 - \alpha_B) \nabla\alpha_A + (2\alpha_B - \alpha_A) \nabla\alpha_B .
\end{aligned}$$

La dérivée de la fonction de poids peut alors être bornée par :

$$||\nabla\alpha|| \leq \sup|(1 - \alpha_B)| \lambda_{\alpha_A} + \sup|(2\alpha_B - \alpha_A)| \lambda_{\alpha_B}$$

Comme on sait que $0 \leq \alpha_B \leq 1$, on peut simplifier l'inégalité en :

$$||\nabla\alpha|| \leq \lambda_{\alpha_A} + \sup|(2 - \alpha_A)| \lambda_{\alpha_B}$$

Opérateur de dépôt.

Soit A et B les deux sous-nœuds de l'opérateur de remplacement qui peut s'écrire :

$$f = f_A + \alpha_B f_B, \qquad \alpha = \alpha_A .$$

Le domaine de définition correspond à $\Omega_0 = \{ \mathbf{p} \in \mathbf{R}^2 \mid \alpha_A(\mathbf{p}) > 0 \}$.

Soit λ_A et λ_B les constantes de Lipschitz de f_A et f_B. La constante de Lipschitz λ de f est définie par $\lambda = \sup_{\mathbf{p}\in\Omega} ||\nabla f(\mathbf{p})||$. Le gradient ∇f peut s'écrire comme suit :

$$\begin{aligned} \nabla f &= \nabla(f_A + \alpha_B f_B) \\ &= \nabla f_A + \nabla(\alpha_B f_B) \\ &= \nabla f_A + \alpha_B \nabla f_B + f_B \nabla \alpha_B . \end{aligned}$$

L'expression ∇f_A est bornée par λ_A, alors que $\alpha_B \nabla f_B$ est bornée par $\sup(\alpha_B)\lambda_B$ et que $f_B \nabla \alpha_B$ l'est par $\sup(f_B)\lambda_{\alpha_B}$. Normalement, $\alpha_B \in [0,1]$ (la valeur peut être ramenée dans l'intervalle sans changer la signification de l'opérateur) on peut en déduire que $\sup(\alpha_B) = 1$ ce qui simplifie l'expression :

$$||\nabla f|| \leq \lambda_A + \lambda_B + \sup_{\mathbf{p}\in\Omega}(f_B)\,\lambda_{\alpha_B}$$

Il est également nécessaire de calculer une borne de Lipschitz pour la fonction de poids. Quand $\alpha = \alpha_A$ le résultat est trivial, mais dans le cas où $\alpha = \alpha_A + {\alpha_B}^2$ il est nécessaire de calculer une nouvelle constante. Le gradient $\nabla\alpha$ peut s'écrire :

$$\begin{aligned} \nabla\alpha &= \nabla(\alpha_A + {\alpha_B}^2) \\ &= \nabla\alpha_A + \nabla({\alpha_B}^2) \\ &= \nabla\alpha_A + 2\alpha_B \nabla\alpha_B . \end{aligned}$$

La dérivée de la fonction de poids peut alors être bornée par :

$$||\nabla\alpha|| \leq \lambda_{\alpha_A} + 2\sup_{\mathbf{p}\in\Omega}(\alpha_B)\,\lambda_{\alpha_B} .$$

Comme $0 \leq \alpha_B \leq 1$, cette expression peut se simplifier en :

$$||\nabla\alpha|| \leq \lambda_{\alpha_A} + 2\lambda_{\alpha_B} .$$

4.3 Résultats

Nous utilisons la bibliothèque écrite en C++ qui permet de manipuler notre modèle de représentation. Deux types de modules de visualisation ont été implémentés : un module CPU pour générer des maillages quelconques (grilles, grilles non-uniformes, maillages adaptatifs construits par triangulations de Delaunay) et les raffiner grâce à un système de découpe et de subdivisons. Il est possible de visualiser les terrains sans *shading* dans une application ou de les

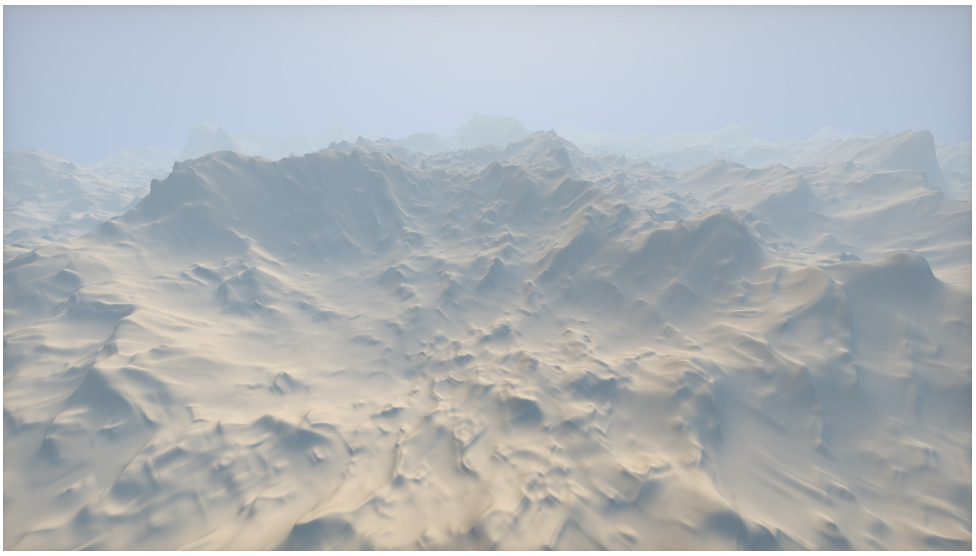


FIGURE 85 : La visualisation par *ray marching / sphere tracing* est plus lente que la rasterisation. Aucune gestion de texture ou de décoration n'est prise en compte dans ces images.

exporter dans EON-VUE XSTREAM® pour les décorer avec une bibliothèque de modèles de végétations et de textures procédurales. Afin d'obtenir des rendus de qualité, nous pouvons importer ces scènes décorées dans MENTALRAY®.

Un prototype de visualisation temps-réel codé en C++ avec la bibliothèque OpenGL/GLSL 4.x permet de modifier directement une scène décrite par un arbre de construction. L'ensemble des images avec un *shading* simple de couleur brune (Fig. 85, Fig. 86) ont été réalisées par ce module. Ce dernier permet de visualiser un modèle de terrains par deux moyens : soit par la création d'un maillage au *runtime* (c'est-à-dire à chaque frame) dont les altitudes des sommets sont calculées sur GPU ; soit par un algorithme de lancer de rayons de type *ray marching / sphere tracing*, lui aussi couplé à l'évaluation d'un arbre de construction sur GPU. L'ensemble des calculs sur CPU est fait sur un ordinateur de bureau équipé d'un INTEL® Core *i*7 avec une fréquence de 3 GHz et 16 Go de RAM et les évaluations GPU ont été réalisées sur GLSL 4.4 sur une NVIDIA® GeForce GTX 670.

4.3.1 Analyse qualitative

La visualisation du modèle de représentation de terrains par arbres de construction peut se faire selon 3 méthodes principales.

Dans le cas d'une visualisation très rapide du relief, sans besoin de décoration, le plus simple est de générer les maillages sur GPU au *runtime* et de calculer l'altitude de chacun des sommets par une évaluation GPU de l'arbre de construction. Cette méthode permet de visualiser en temps-réel un terrain et autorise une modification interactive du relief (construction de l'arbre, changement des paramètres de nœuds).

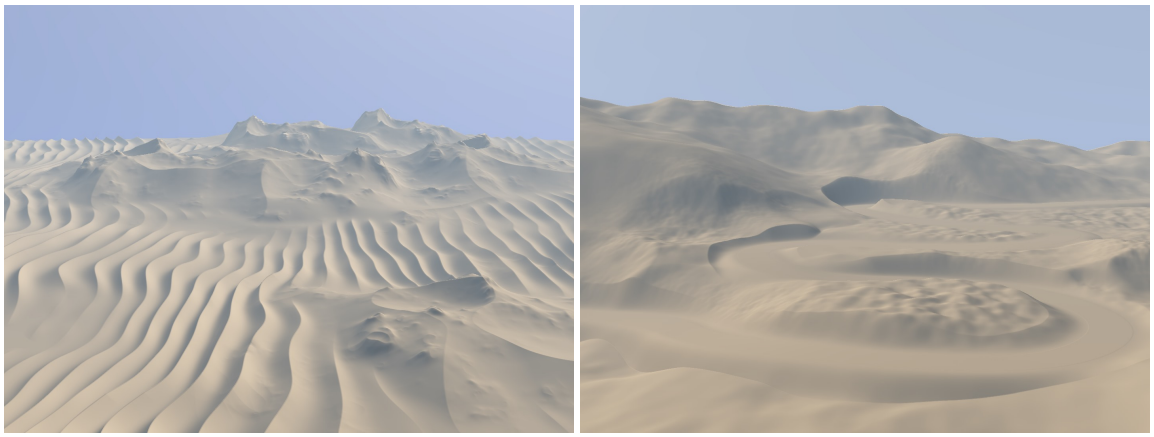


FIGURE 86 : Exemples de terrains rendus en temps-réel sur GPU avec l'algorithme de *sphere tracing*. Cette méthode permet de visualiser rapidement et sans artéfact des géométries fines et saillantes comme les crêtes des dunes de sables.

Dans le cas d'une visualisation de type production, il est préférable de calculer un maillage raffiné sur CPU. Ce maillage peut être obtenu par des procédures simples (grilles, primitives géométriques simples) ou par la construction d'un nuage de points adaptatif guidé par des fonctions d'importances et donnant au final une triangulation de Delaunay. Ce processus de visualisation est plus lent, mais permet d'extraire des maillages de qualité dont les bords sont bien découpés et de l'importer dans un logiciel de décoration de scènes naturelles comme EON-VUE XSTREAM®. La scène complète peut ensuite être rendue dans un moteur de jeux-vidéo temps-réel (comme UNITY® ou l'UNREAL ENGINE®) ou à travers un logiciel spécialisé dans le rendu de haute qualité par lancer de rayons, comme MENTALRAY®.

Enfin, une perspective intéressante consiste à visualiser le modèle de représentation de terrains par arbre à l'aide d'un lancer de rayons sur GPU. Contrairement à la génération de maillages réguliers ou adaptatifs sur GPU, qui fait intervenir deux discrétisations de la fonction continue représentée par le modèle d'arbre (une selon le plan 2D Oxy, puis une durant la rasterisation), les méthodes à base de lancer de rayons ne font qu'une seule approximation (l'échantillonnage selon l'espace écran). On visualise d'une façon plus exacte la fonction continue représentant le terrain. En particulier, les reliefs discontinus non C^1 (*e. g.* les crêtes ou les dunes de sables) sont bien mieux échantillonnés quelle que soit la position de la caméra. Notre prototype de visualisation peut se faire en temps interactif sur des scènes de petites tailles, à l'aide d'un algorithme de type *ray marching / sphere tracing* ; de nombreuses optimisations sont encore possible pour pouvoir gérer des scènes plus complexes.

4.3.2 Performances

Notre modèle repose sur un nombre de primitives et d'opérateurs génériques relativement réduit. Les fonctions d'élévation $f(\mathbf{p})$ et de poids $\alpha(\mathbf{p})$ sont suffisamment simples pour que leur calcul soit transposable dans un *shader program*. Les paramètres caractéristiques d'un arbre de

Tableau 3 : L'algorithme de *sphere tracing* produit des images à la résolution FullHD (1920 × 1080). Les terrains provenant du Chapitre 5 n'ont pas été rendus avec les techniques GPU.

Scène	Maillage raffiné (CPU)		*Sphere tracing* (GPU)		Maillage régulier (GPU)	
	# $f(\mathbf{p})$	⏲ (ms)	# $f(\mathbf{p})$	⏲ (ms)	# $f(\mathbf{p})$	⏲ (ms)
Fig. 46	–	–	55 M	36	5 M	6
Fig. 68	–	–	50 M	60	20 M	25
Fig. 70	–	–	79 M	561	20 M	150
Fig. 71	–	–	66 M	280	80 M	320
Fig. 75	–	–	60 M	182	5 M	14
Fig. 87	–	–	52 M	361	80 M	560
Fig. 115	1 M	1000	–	–	–	–
Fig. 117	1 M	600	–	–	–	–
Fig. 118	1 M	1000	–	–	–	–
Fig. 119	1 M	800	–	–	–	–

construction et les paramètres des différents nœuds de l'arbre sont stockés dans deux *shader storage buffers objects* (SSBO). L'évaluation d'une fonction se fait en analysant le premier SSBO contenant la structure de l'arbre et en appelant les fonctions génériques avec leurs données correspondantes.

Nous avons implémenté deux techniques différentes de visualisation de nos terrains sur GPU : un algorithme de *sphere tracing* (Section 4.2) et une tesselation adaptative basée sur un *quadtree* comme présenté par Dupuy *et al.* [105]. Les deux méthodes utilisent le GPU pour évaluer l'arbre de construction représentant le terrain et permettent de générer des images en un temps interactif (Fig. 85).

La tesselation adaptative basée sur un *quadtree* permet de visualiser le terrain avec une résolution multi-échelles et sans trous apparents en utilisant un critère de niveau de détails qui dépend de la distance à la caméra. Cette technique tire avantage de la tesselation matérielle et des capacités de *culling* du GPU.

Il est préférable d'utiliser un algorithme de tessellation parce qu'il est rapide, simple à comprendre, et nécessite un nombre prévisible d'évaluations. Le temps de rendu est alors dépendant de la complexité de l'arbre de construction. A l'inverse, le *sphere tracing* génère des images de meilleure qualité en particulier sur les géométries saillantes et fines. La méthode peut également inclure facilement un calcul des ombres. Si le nombre de pixels est constant, le nombre d'évaluations de l'arbre (par rayon) peut être élevé et dépend fortement du placement et de l'orientation de la caméra.

Dans la plupart des cas, l'évaluation d'un arbre de construction pour visualiser un terrain prend quelques secondes sur CPU. Le modèle de représentation étant hautement parallèle (*embarrassingly parallel*), l'utilisation du GPU permet de descendre les temps de calculs à quelques millisecondes ce qui autorise une exploration/édition interactive des scènes. Le Tableau 3 présente quelques statistiques sur la visualisation des différentes scènes illustrant l'ensemble du chapitre. Pour les scènes composées de plusieurs centaines de primitives, l'élagage spatial de l'évaluation (grâce à une hiérarchie de boîtes englobantes) s'avère très utile pour diminuer le nombre de calculs à faire. L'élagage fréquentiel permet d'éviter encore plus de calculs et accélère l'algorithme de *sphere tracing* en augmentant les pas et en diminuant les constantes de Lipschitz.

4.4 Limitations et perspectives

Nous présentons certaines limitations et perspectives qui touchent à la visualisation du modèle de représentation de terrains par arbres de construction.

4.4.1 Limitations

La visualisation du modèle est réalisée selon plusieurs méthodes différentes. Actuellement, la visualisation GPU a été réalisée au sein d'un prototype qui permet à la fois de rasteriser un terrain en temps-réel et de calculer une image avec un lancer de rayons. Dans tous les cas, il serait **nécessaire de découpler l'évaluation d'un arbre de construction** sur les sommets d'un maillage, **de la visualisation proprement-dite** de ce même maillage. Pour l'instant, ces deux étapes sont intrinsèquement liées au sein d'une architecture conçue avec OpenGL qui empêche d'utiliser le GPU simplement comme un processeur hyper-parallèle.

Le modèle GPU est également limité techniquement par les capacités matérielles d'une carte graphique : jusqu'à aujourd'hui il est encore difficile de gérer efficacement des très grandes scènes (comme dans les îles générées dans le Chapitre 5).

Certaines des étapes CPU sont également difficile à paralléliser sur GPU : la construction d'un maillage adaptatif par un algorithme de Delaunay ou le raffinement précis des triangles sont des étapes complexes à écrire de manière parallèle. Il est donc impossible pour l'instant d'avoir des mécanismes de visualisation temps-réel sur GPU de terrains multi-matériaux quand on utilise un *pipeline* centré sur la rasterisation (Fig. 87).

Enfin, une limitation intrinsèque du modèle provient de la difficulté à représenter les surplombs. Le modèle permet tout de même de représenter des couches de matériaux superposées et il est possible de considérer certaines de ces couches comme du vide ou de l'air. Malgré tout, il est difficile de visualiser ces couches : pour extraire un maillage il faudrait générer les maillages fermés (orientés et sans bord) de chaque couche de matériaux. Pour visualiser en temps-réel, il serait probablement nécessaire de passer par un algorithme d'extraction d'isosurface 3D type *marching cube*.

FIGURE 87 : Exemple de route serpentant à travers la montagne. L'artiste a esquissé la scène en mélangeant une centaine de primitives disques pour définir le relief. La trajectoire de la route a été définie à l'aide d'une vingtaine de courbes quadriques et l'opérateur de remplacement a effectué le terrassement. La couche d'asphalte de la route est modélisée avec un arbre de construction et le bord du domaine est découpé proprement avec l'algorithme de raffinage de triangles.

4.4.2 Perspectives

Il serait intéressant dans un premier temps de **découpler la visualisation de l'évaluation de l'arbre**. La conception d'un programme GPGPU capable d'évaluer un arbre de construction sur un tableau de données (correspondant à un ensemble de points du plan Oxy) permettrait un calcul rapide et efficace quel que soit le pipeline de visualisation du terrain.

Actuellement, les arbres de construction sont évalués en profondeur d'abord (*depth first*) ; la parallélisation se faisant alors sur n évaluations d'un même arbre de construction. Comme chaque arbre possède automatiquement une hiérarchie de boîtes englobantes, les n évaluations correspondent potentiellement à différents chemins dans l'arbre. Une perspective intéressante serait de tester un mode de calcul en largeur d'abord (*breadth first*) qui permettrait de calculer chaque primitive instanciée une seule fois avec un mécanisme de cache. La parallélisation s'effectuerait en calculant à chaque fois les n valeurs correspondant à un unique nœud (il n'y aurait qu'un seul parcours d'arbre et tous les processeurs feraient la même opération).

Le modèle de représentation ne permet pas non plus de **gérer une notion de texture**. Si le stockage de la paramétrisation des textures dépend du modèle, une alternative serait d'utiliser des textures volumiques qui seraient calculer à la visualisation. Comme les textures peuvent aussi représenter des détails à la surface (*normal maps*, *displacement maps*), il serait nécessaire d'avoir des garanties sur la géométrie finale (pas de morceaux de surfaces flottantes).

Nous avons proposé un système de niveaux de détails (Section 3.5), et un système de raffinement de maillage (Section 4.1) : il serait intéressant de coupler de façon efficace les deux pour obtenir un mécanisme de raffinement et de dé-raffinement dépendant de la position de la caméra.

Enfin, **le modèle de relief devrait être filtrable**. Cela permettrait d'observer le modèle, quelle que soit la distance et quel que soit l'angle de vue, et de pouvoir obtenir une projection fidèle à l'écran de la fonction continue encodée par l'arbre de construction. Filtrer revient à calculer l'intégrale des contributions visuelles du relief dans un pixel donné ; ce calcul dépend non seulement de la géométrie (et de l'éclairage) mais aussi des textures plaquées sur le modèle.

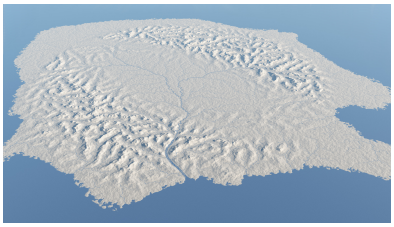
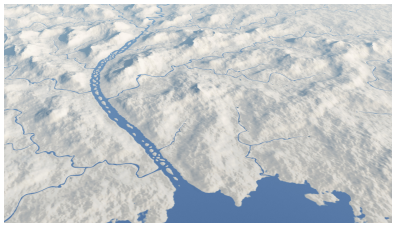
4.5 Conclusions

Nous avons proposé **plusieurs méthodes de visualisation** du modèle de représentation de terrains par arbres de construction. Chaque méthode possède des avantages et des défauts : les évaluations GPU (par rasterisation ou par lancer de rayons) permettent une interaction temps-réel avec le terrain mais n'autorisent pas la conception de très grandes scènes ni leur décoration. Les propriétés mathématiques des bornes de Lipschitz que nous pouvons calculer sur l'ensemble de nos terrains, permettent l'utilisation d'un algorithme de *sphere tracing* qui garantit de trouver correctement les intersections entre un rayon et le terrain. À contrario, la visualisation CPU permet d'exporter des maillages pour les décorer et les visualiser avec un rendu de très bonne qualité.

Il est possible d'étendre les fonctionnalités de notre modèle pour permettre **une visualisation de meilleure qualité et plus rapide**. En particulier, un système de découpe et de raffinement/déraffinement compatible avec une visualisation GPU serait très utile pour modifier les terrains en temps-réel.

5

Génération procédurale de terrains


Sommaire

Science, like art, is not a copy of nature
but a re-creation of her.

— Jacob BRONOWSKI

Dans ce chapitre, nous présentons un algorithme de génération procédurale de terrains. Cet algorithme cherche à créer des îles réalistes en générant leur réseau hydrographique. Ces réseaux hiérarchiques structurent le terrain en créant des vallées et des lignes de crêtes. L'algorithme s'appuie sur les travaux réalisés en géomorphologie, en s'inspirant en particulier de la notion de nombre de Horton-Strahler et de la classification de Rosgen. À l'aide de plusieurs méthodes *ad hoc*, nous générons des informations de haut niveau sur la forme des cours d'eau, leur débit ou leur trajectoire. Ces données sémantiques sont converties dans un arbre de construction correspondant au formalisme présenté Chapitre 3 et qui représente la fonction continue d'élévation décrivant le relief. Des informations supplémentaires concernant la distribution de la végétation ou l'épaisseur de la couche d'eau sont également calculées. Le modèle 3D généré est finalement extrait et importé dans un logiciel de rendu.
Après avoir rappelé quelques principes de géomorphologie (Section 5.1), nous présentons le *pipeline* général de l'algorithme de génération procédurale d'îles (Section 5.2). Cette méthode peut être décomposée en trois grandes étapes qui seront, chacune, décrites en détails : la génération du réseau hydrographique (Section 5.3), la génération d'informations sémantiques (Section 5.4) et enfin la génération automatique du modèle de terrain représenté par un arbre de construction décrivant le relief du terrain (Section 5.5). Après avoir présenté plusieurs exemples d'îles générées et analysé les temps de génération (Section 5.6), nous détaillons les principales limitations et perspectives (Section 5.7) de ce travail avant de conclure ce chapitre (Section 5.8).
Ces travaux ont fait l'objet de deux publications [139, 140].

5.1 Bases d'hydrologie

L'eau a un impact majeur sur la formation des reliefs. Le cycle de l'eau structure le terrain à l'échelle régionale et forme un réseau de vallées qui permet un écoulement de l'eau jusqu'à la mer. Les mouvements de l'eau creusent également directement les sols (plus ou moins facilement selon leur composition) ce qui engendre la création de rivières.

On parle d'hydrologie (« *hydro* » venant étymologiquement du grec « *hudôr* » c'est-à-dire « *eau* ») quand on étudie le cycle de l'eau et ses échanges avec la surface terrestre ou l'atmosphère. Les notions de réseau hydraulique (combinaison de la racine « *hydr-* » et du mot grec « *aulos* » c'est-à-dire « *tuyau* ») et de réseau hydrographique font référence au transport de l'eau sur une surface ou en sous-sol. Enfin, l'érosion hydrique est l'ensemble des effets que provoque le transport de l'eau sur un relief.

Nous présentons trois notions importantes que nous avons pris en compte pour la conception de l'algorithme de génération procédurale d'île avec écoulement et sans simulation. Les bassins-versants (Section 5.1.1) permettent de structurer le terrain à grande échelle. L'algorithme de Horton-Strahler (Section 5.1.2) permet de mesurer la complexité d'un réseau hydrographique en observant la topologie de l'arbre d'écoulement. Enfin, la classification de Rosgen (Section 5.1.3) décrit visuellement les cours d'eau et permet de définir un ensemble d'archétypes de rivières.

En complémenter de cette section, nous invitons le lecteur à consulter plusieurs cours d'hydrologie disponibles sur Internet [114, 149, 169].

5.1.1 Bassins-versants

Un bassin-versant, noté W, est un territoire qui draine l'ensemble de ses eaux (provenant généralement de précipitations ou de fontes de neiges) vers un exutoire commun. On calcule généralement le bassin-versant correspondant à une embouchure ou à un point se situant sur une rivière. Les bords d'un bassin-versant sont appelés des lignes de partage des eaux : deux gouttes de pluie qui tombent de part et d'autre de cette ligne n'emprunteront pas la même trajectoire et ne s'écouleront pas vers le même exutoire. En ignorant la perméabilité des sols et en admettant que le chemin d'une goutte est calculé de manière déterministe (subissant la gravité et guidé par les pentes du relief), les lignes de partage des eaux correspondent aux lignes de crêtes.

Chaque bassin est généralement composé d'une infinité de sous-bassins (une fois l'eau écoulée jusqu'à ces sous-exutoires, elle continue son chemin jusqu'à l'exutoire du bassin principal). Ainsi, le bassin-versant d'un point **p** contient le bassin-versant d'un point **q** si celui-ci est situé en amont de **p** (Fig. 88). Si on établie un lien entre bassin-versant et rivière, on peut en déduire cette propriété : si un petit cours d'eau R_j se jette dans un fleuve R_i, alors le bassin-versant du cours d'eau en amont $W(R_j)$ est contenu dans le bassin-versant du fleuve $W(R_i)$.

$$R_i \xleftarrow{\text{est en aval}} R_j \Rightarrow W(R_i) \supset W(R_j)$$

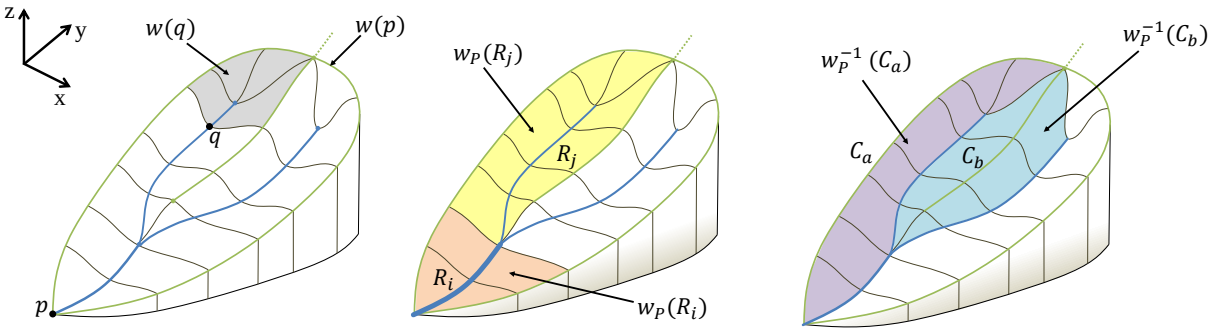


FIGURE 88 : Les bassins-versants structurent le paysage ; ils sont généralement orientés vers les talwegs et leurs frontières correspondent souvent à des lignes de crêtes. La notion de bassin-versants propres (rattachés à une rivière ou à une crête) permet de partitionner l'espace.

Cette définition pose tout de même problème : les minima locaux du terrains (par exemple, un affaissement de quelques centimètres) créent des bassins-versants qui ne sont pas reliés dans la hiérarchie. Il est possible de prendre en compte ces dépressions internes en simulant leur « remplissage » [144], méthode très utile pour calculer l'ensemble des écoulements sur un terrain comme dans les algorithmes D8 [269] et D-inf [368].

Quand on manipule un graphe correspondant au réseau hydrographique, VIMONT [395] propose de définir la notion de bassin-versant propre à une rivière, noté $W_{\mathcal{P}}(R_i)$, comme la zone de terrain où les eaux de pluie vont directement s'écouler dans le cours d'eau en question sans passer au préalable par un affluent de cette rivière (Fig. 88 centre). Cette définition permet de construire un partitionnement du domaine : le bassin-versant d'un fleuve correspondra à l'union de son bassin-versant propre et des bassins-versants de ses affluents (et ceci récursivement).

En réalité, comme il n'y a pas forcément d'écoulement d'eau permanent, l'ensemble des points dont l'altitude est la plus basse (*i. e.* le fond d'une vallée) est appelé talweg. Ce réseau hiérarchique de talwegs structure l'ensemble du relief. Le dual de ce réseau correspond au réseau des crêtes. Une ligne de crête séparant deux talweg est appelé interfluve. Par conséquent, le partitionnement en bassins-versants propres proposé par VIMONT peut être étendu sur le réseau dual (Fig. 88 droite). Ces bassins-versants propres de crêtes sont notés $W_{\mathcal{P}}^{-1}(C_i)$.

5.1.2 Nombre de Horton-Strahler

Pour structurer un réseau hydrographique, les nombres de Horton-Strahler (*Horton-Strahler number* ou *Strahler stream order*, proposé par STRAHLER en 1952 [358]) permettent de définir une hiérarchie d'affluents dans un réseau hydraulique. Ce nombre est calculé par un algorithme relativement simple. Prenons le cas d'un arbre représentant un réseau hydrographique ; le nombre de Horton-Strahler d'une arête de cet arbre correspond au nombre du nœud en amont :

- le nombre d'Horton-Strahler d'une feuille est toujours $s = 1$;
- si un nœud interne (*i. e.* une confluence) a un enfant dont le nombre de Horton-Strahler est s_i et que tous ses autres enfants ont un nombre de Horton-Strahler s_j tel que $s_j < s_i$, alors le nombre de Horton-Strahler de la confluence reste $s = s_i$;

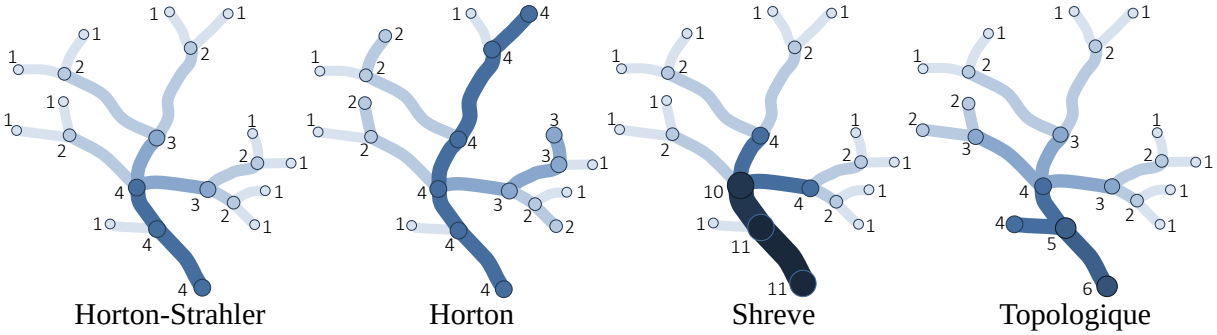


FIGURE 89 : De nombreuses méthodes existent pour mesurer la complexité d'un réseau hydrographique. La méthode de Horton-Strahler permet de faire ressortir des jonctions symétriques et des jonctions asymétriques.

– si un nœud interne (*i. e.* une confluence) a au moins deux enfants dont le nombre de Horton-Strahler est s_i et que tous ses autres enfants ont des nombres de Horton-Strahler $s_j \leq s_i$, alors le nombre de Horton-Strahler de la confluence augmente tel que $s = s_i + 1$.

Cet algorithme différencie deux types de jonctions : celles qui augmentent l'indice de Horton-Strahler et que nous appelons jonctions symétriques, et les autres que nous appelons jonctions asymétriques. Un nœud de complexité i a forcément au moins deux descendants de complexité $i - 1$ qui ont eux même récursivement chacun deux enfants (donc 4 nœuds) de complexité $i - 2$, ... Un arbre qui contient n nœuds peut avoir au plus une complexité de Horton-Strahler de $\log_2 n$ (cas d'un arbre binaire complet). On considère généralement que le nombre de Horton-Strahler d'un arbre correspond au nombre contenu à la racine. Sur Terre, le plus grand nombre de Horton-Strahler est 12 et correspond à l'Amazonie mais la très grande majorité (près de 80%) des cours d'eau ont un indice de complexité entre 1 et 3 [57].

Le nombre de Horton-Strahler peut servir à d'autres fins que l'hydrologie : il est par exemple utile pour calculer le nombre de registre minimal pour effectuer une opération arithmétique (ce calcul est utilisé incidemment dans l'évaluation GPU du modèle d'arbre de construction présenté Chapitre 4).

D'autres moyens de mesurer la complexité d'un réseau hydrographique existent. La méthode originelle de Horton attribue un indice de complexité à l'ensemble de la rivière (et pas seulement à un segment de la rivière) : le nombre de Horton pour le Rhône est de 10 depuis sa source près de Genève jusqu'à son embouchure alors que le nombre de Horton-Strahler est de 10 uniquement sur le segment entre Lyon et l'embouchure. La méthode originelle de HORTON a été abandonnée au profit des nombres de Horton-Strahler pour éviter de choisir – de manière souvent subjective – une rivière principale. Un autre algorithme très utilisé, proposé par SHREVE, consiste à additionner les complexités à chaque jonction. Il existe de nombreuses autres méthodes pour mesurer la complexité d'un réseau (Fig. 89) ; des algorithmes plus classiques comme la mesure de la profondeur topologique d'une rivière dans un réseau sont également utilisables.

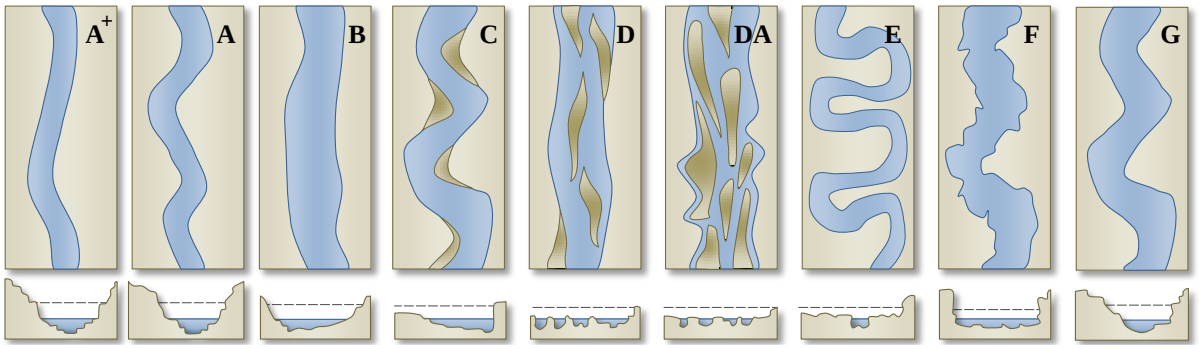


FIGURE 90 : La classification de Rosgen permet de décrire visuellement les cours d'eau et en particulier leur trajectoire, leur profil transversal, et la composition géologique de leur sol.

5.1.3 Classification de Rosgen

La classification de Rosgen [326], proposée publiquement en 1994 dans le prestigieux journal CATENA puis développée dans le livre *Applied River Morphology* [327] en 1996, définit 9 archétypes de cours d'eau. Ce modèle permet de classer les rivières selon des critères presque exclusivement visuels (*i. e.* sans beaucoup de références à des données physiques). Chaque archétype – identifié par une lettre – correspond à un type de trajectoire et de profil transversal clairement identifiables (Fig. 90). Ainsi, les types **A**$^+$, **A**, **B** sont des cours d'eau de montagnes dont la pente est forte. Le type **G** correspond à des rivières encaissées alors que les rivières de type **F** sont très larges et très peu profondes. Les rivières de type **C** et **E** sont très sinueuses avec de nombreux méandres. Enfin, les types **D** et **DA** représentent les rivières en forme de tresses. La classification intègre également le contexte géologique de chaque type de rivière (présence de roches, de pierres et galets, de gravier, sable, limon ou encore argile). Les transitions entre deux types de Rosgen sont relativement peu étudiées (Fig. 91).

Pour définir ou construire un type de cours d'eau précis, les ingénieurs utilisent un graphe de décision qui fait intervenir plusieurs critères décisifs : la topologie de la rivière (simple ou en tresse), son coefficient d'encaissement dans le sol, le ratio entre sa largeur et sa profondeur moyenne, sa sinuosité, et son coefficient de pente. Certains de ces critères sont liés (il n'existe

FIGURE 91 : Les transitions entre les types sont encore peu étudiées (ici, passage d'un type **E** vers **D**).

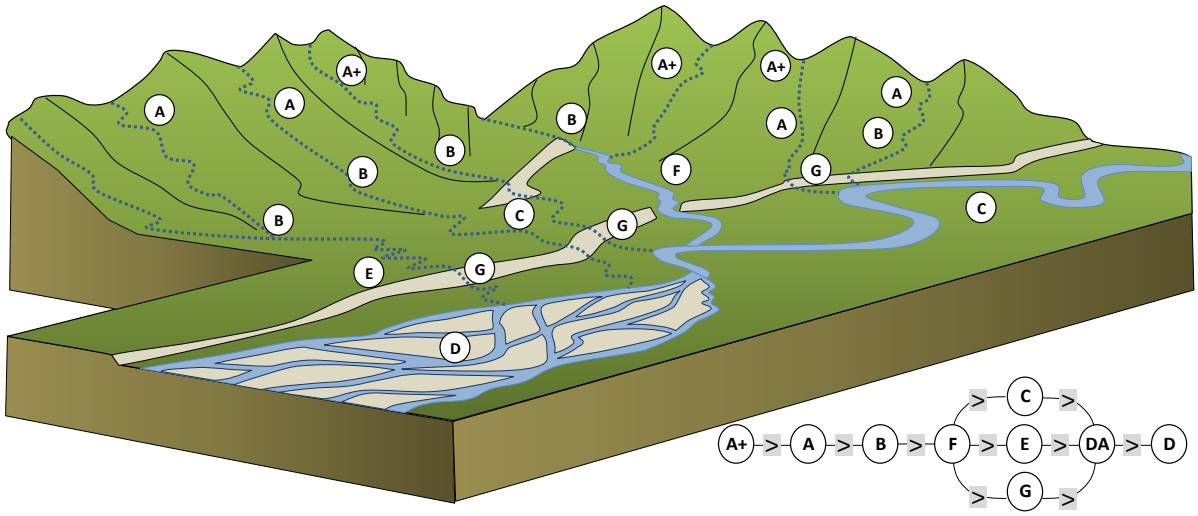


FIGURE 92 : La classification de Rosgen définit une relation d'ordre partiel de quel type de cours d'eau peut se jeter dans quel type. Ainsi, certains types (**A**$^+$, **A**, **B**) sont majoritairement localisées aux niveaux des sources montagneuses alors que d'autres (**D**, **DA**) sont proches des embouchures de fleuves.

par exemple aucune rivière en tresse avec une pente à 10%) sans que l'on puisse pour autant connaître toutes leurs interdépendances. Il est relativement aisé de trouver ces matrices de décision sur Internet [325].

ROSGEN ne propose à priori aucun ordre d'écoulement pour ces différents types de rivières (*e. g.* tel type de cours d'eau est forcément en aval de tel autre type) ; on peut pourtant conjecturer qu'il existe un ordre partiel naturel (Fig. 92). Les sources des rivières ayant tendance à être placées dans les zones montagneuses, les ruissellements sont généralement les plus forts en amont. Il existe certainement des exceptions, par exemple sur les hauts plateaux. Dans le cadre de l'utilisation de nos algorithmes de génération procédurale, nous simplifions la classification de Rosgen pour utiliser l'ordre partiel indiqué Fig. 92.

La classification de Rosgen a été critiquée notamment parce qu'elle ne prend pas en compte les caractéristiques physiques de taille des rivières. Il est difficile de prévoir le nouveau état d'équilibre dans lequel se trouvera un cours d'eau ayant subi un changement important et qui sortirait ainsi de sa catégorie de départ. Pour notre part, comme nous nous intéressons avant tout à une génération phénoménologique et que nous ne faisons pas intervenir de simulations d'érosion, les défauts de la classification de Rosgen nous apparaissent suffisamment minimes pour les ignorer dans un premier temps.

5.2 Vue générale de l'algorithme

L'algorithme de génération procédurale de terrains repose sur un constat : il n'existe pas de méthode de génération *ex-nihilo* de terrains érodés (c'est-à-dire comprenant des vallées et des lignes de crêtes). La solution usuelle consiste à générer procéduralement un terrain fractal puis à

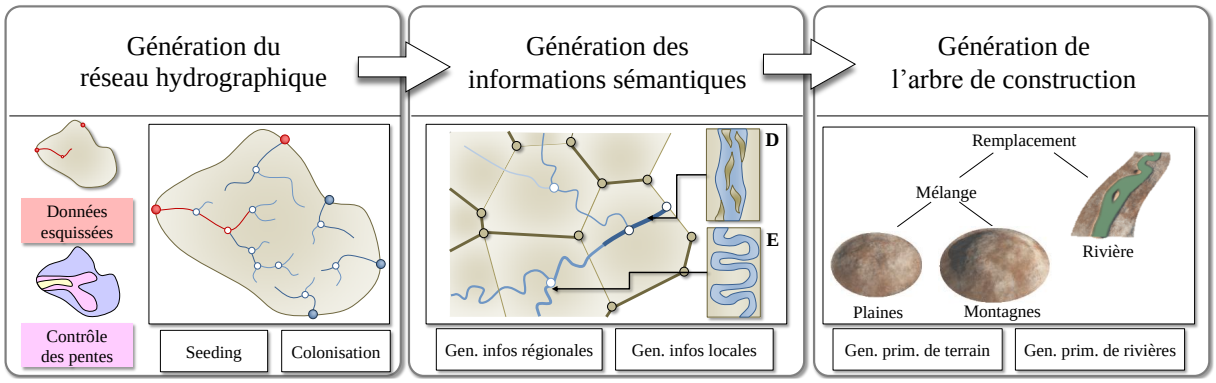


FIGURE 93 : Vue générale du *pipeline* de l'algorithme de génération de terrains. Le système génère automatiquement un réseau hydrographique à partir d'un contour dessiné par l'utilisateur. Une fois le graphe du réseau construit, on génère des informations sémantiques supplémentaires (bassins-versants, débits, trajectoires précises, ...). Finalement, un arbre de construction est généré en plaçant un ensemble de primitives paramétrées selon la nature du terrain. Le modèle de terrain peut alors être visualisé.

appliquer des simulations d'érosions. Cette solution, déjà coûteuse en temps de calcul, ne donne pas entière satisfaction, car si elle améliore localement le réalisme du relief, elle ne crée aucune structure (*e. g.* un terrain fractal érodé donne une série de lacs de montagne se ressemblant tous).

Dans un paysage réel, les phénomènes d'érosion structurent le terrain en bassins-versants et c'est cette structuration qui est absente à la fois dans les méthodes de génération à base de fractales, dans celles à base de subdivision, et dans les simulations d'érosion. Pour obtenir une telle structuration, il est possible de modéliser les résultats de ces érosions (par exemple à l'aide d'outils d'esquisse). Nous proposons de générer automatiquement cette structuration sans esquisse, ni aucune simulation physique. Nous souhaitons également donner un contrôle de haut niveau aux utilisateurs de la méthode qui permette de guider la structuration des paysages. Enfin, un autre objectif est de rendre les cours d'eau visuellement riches et complexes au moment de la construction de leur géométrie.

Le *pipeline* de l'algorithme est articulé autour de trois étapes distinctes (Fig. 93).

Au cours de la première étape (Section 5.3), nous cherchons à **générer un réseau hydrographique**. Pour commencer, l'utilisateur dessine interactivement un contour définissant un domaine dont il souhaite générer le terrain. Il peut également placer manuellement des embouchures de fleuves sur ce contour, esquisser des rivières qui apparaîtront sur le terrain final et contrôler certains des paramètres globaux de génération en fournissant des cartes de contrôles qui vont contraindre la génération du réseau et du relief. À partir de ces entrées, le système va générer un réseau hydrologique. Ce réseau, qui couvre l'ensemble du domaine que l'utilisateur veut générer, est représenté par un graphe topologique plongé géométriquement dans un espace à deux dimensions. L'algorithme d'expansion représentant la conquête du réseau hydrographique, est codé sous la forme d'une grammaire s'inspirant de l'algorithme de Horton-Strahler qui permet de quantifier la complexité d'une structure arborescente. Au fur et à mesure que le graphe est

construit, les nœuds représentant les rivières se voient attribuer une altitude qui garantit un écoulement depuis les sources d'eau jusqu'aux embouchures.

La seconde étape (Section 5.4) consiste – une fois le réseau hydrographique complément défini – à **générer des informations sémantiques** qui décrivent la nature du terrain et des cours d'eau. Pour cela, on va extraire la topologie et la géométrie du graphe pour pouvoir décomposer le domaine en un ensemble de cellules de Voronoï. Cette décomposition hiérarchique correspond aux bassins-versants et sous-bassins-versants et permet de calculer de nouvelles informations très précieuses comme le débit en tout point du réseau hydrographique. Durant cette étape, nous étiquetons également chaque cours d'eau selon la classification de Rosgen. Une fois ces labels attribués, nous pouvons raffiner les trajectoires de ces rivières et travailler spécifiquement sur les jonctions, les sources ou encore les deltas. Au cours de cette étape, une élévation est également attribuée aux points de crêtes (correspondant aux sommets des cellules de Voronoï). Celle-ci est calculée à partir d'une carte de contrôle donnée par l'utilisateur qui pourra agir en indiquant si une zone est montagneuse ou plutôt plate.

Enfin, au cours de la troisième et dernière étape du processus (Section 5.5), nous cherchons à **générer un modèle de terrain** représenté par un arbre de construction décrivant le relief. Cette structure représente la fonction continue, sans surplomb, d'élévation de l'intégralité de l'île générée ; cette fonction permet d'encoder le relief de l'île avec une faible empreinte mémoire. Un fois le modèle d'arbre construit, il suffit d'extraire un maillage ou de visualiser la fonction directement.

5.3 Génération du réseau hydrographique

Le réseau hydrographique qui couvre le domaine donné par l'utilisateur en entrée est un graphe géométrique qui est généré lui-même en deux étapes. Une première étape d'amorce consiste à distribuer des nœuds de rivières sur le contour du domaine au niveau des embouchures (Section 5.3.1). D'autres nœuds sont placés sur les rivières construites préalablement par l'utilisateur. Dans un deuxième temps, les rivières sont générées par une colonisation progressive du domaine (Section 5.3.2).

Contrôle utilisateur

Pour amorcer l'algorithme, l'utilisateur peut fournir plusieurs données d'entrée. Un contour polygonal Γ dessiné à la main ou généré automatiquement va délimiter le domaine où l'algorithme génère le réseau. Ce contour décrit généralement un iso-contour du terrain dont l'altitude est nulle, ce qui revient à dire que ce contour délimite les côtes et les plages. L'utilisateur peut également esquisser plusieurs morceaux du réseau hydrographique qui apparaîtront une fois le graphe des rivières construit. Les nœuds que l'utilisateur place (sur la côte ou dans les terres) n'ont pas d'élévation et seront potentiellement connectés à chaque expansion.

Deux cartes de contrôles guidant l'algorithme peuvent également être données par l'utilisateur. L'une d'elles va déterminer l'emplacement des zones montagneuses et des zones plates,

l'autre va guider la construction du réseau hydrographique en agissant sur la pente des cours d'eau. L'utilisateur peut construire ces cartes de contrôle manuellement, ou les générer à l'aide d'une fonction procédurale.

Enfin, les paramètres globaux (*e.g.* influant sur la probabilité d'appliquer une règle de la grammaire plutôt qu'une autre) peuvent être définis par l'utilisateur au début de l'algorithme.

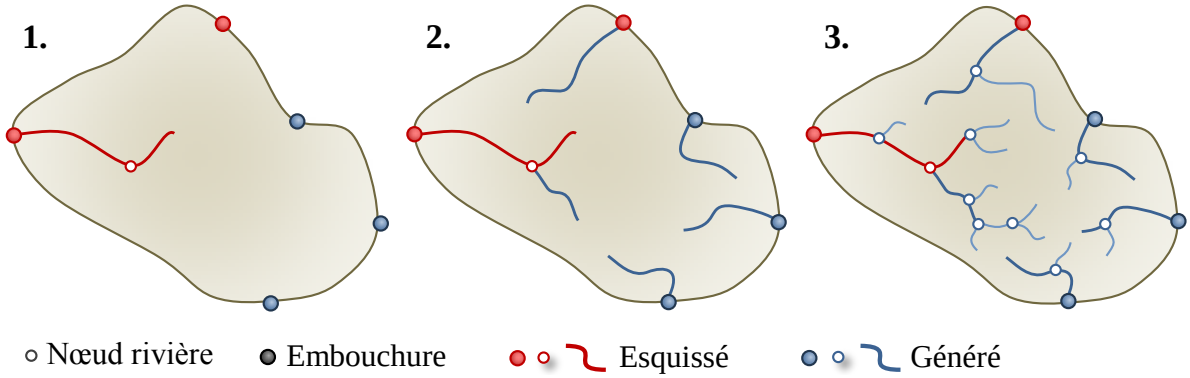


FIGURE 94 : À partir d'un contour initial Γ et d'une esquisse partielle du réseau hydrographique **G** (étape 1), le graphe des rivières est complété par conquêtes géométriques successives jusqu'à avoir recouvert le domaine Ω (étapes 2 et 3).

Notations

Soit Ω le domaine où l'algorithme va générer un réseau et Γ son contour défini par un polygone de contrôle 2D. L'algorithme va créer un ensemble d'arbres noté **G** recouvrant Ω. Un arbre est défini par son ensemble de nœuds-rivière **R** et son ensemble d'arêtes droites **E** qui ont une taille constante e définie au début du processus. Chaque nœud $R_i = (\mathbf{p}_i, s_i, \rho_i, \phi_i)$ a une position $\mathbf{p}_i$, un indice de priorité s_i, un type de rivière ρ_i en accord avec la classification de Rosgen [326], et un débit ϕ_i. Certaines de ces informations seront attribuées lors de la prochaine étape de l'algorithme.

5.3.1 Création initiale des candidats

La première étape consiste en la création des nœuds candidats initiaux qui seront étendus par la suite. Les nœuds candidats sont placés sur le contour Γ au niveau des futures embouchures. Si l'utilisateur a esquissé des morceaux de rivières, des nœuds sont également placés aux extrémités et le long des trajectoires de façon régulière comme montré sur la Fig. 94. Chaque nœud se voit assigné un indice de priorité qui va définir son importance. Ce sont aussi bien les positions que les priorités qui vont donner l'apparence finale du réseau hydrographique construit.

Le placement des nœuds initiaux peut être contrôlé manuellement par l'utilisateur ou généré automatiquement à l'aide d'un ensemble d'heuristiques. Nous intégrons deux conditions *ad hoc* : les embouchures de fleuves importants doivent généralement être séparées et les fleuves ont tendance à se jeter dans la mer au niveau des parties concaves du contour.

5.3.2 Colonisation du domaine

Cette étape de colonisation du domaine Ω permet de construire le réseau hydrographique de façon incrémentale. Au fur et à mesure que le graphe **G** s'étend (en suivant un algorithme probabiliste), ces nouvelles extensions géométriques prennent de l'altitude. Le graphe représentant le réseau possède un ensemble noté **X** de nœuds-candidats à l'expansion qui seront choisi à l'aide d'un algorithme de tri-sélection et qui seront étendus à l'aide de plusieurs règles. Trois étapes sont répétées jusqu'à coloniser l'ensemble du domaine :

1. **Sélection d'un nœud :** on choisit un nœud noté $R_{\mathbf{X}}$ de l'ensemble des nœuds-candidats **X** qui sera étendu.
2. **Choix de l'expansion :** on étend le nœud sélectionné $R_{\mathbf{X}}$ à l'aide d'une grammaire dont les règles de subdivision s'inspirent de l'algorithme de Horton-Strahler.
3. **Expansion géométrique :** une fois une règle d'expansion choisie, on place virtuellement les nouveaux nœuds **N** et on vérifie que la nouvelle configuration est compatible avec le graphe déjà créé puis – si le placement est valide – on met à jour la liste des candidats **X** avec les nouveaux nœuds :

$$\mathbf{X} \leftarrow (\mathbf{X} \setminus \{R_{\mathbf{X}}\}) \cup \mathbf{N}.$$

   Le graphe **G** est également mis à jour pour intégrer les nouveaux nœuds **N**. Si un des nouveaux nœuds n'est pas compatible avec les nœuds déjà présents, il est supprimé du graphe.

Bien que l'algorithme de colonisation s'appuie sur un processus de subdivision défini par une grammaire et s'inspirant de l'algorithme de Horton-Strahler, la méthode de génération de terrains est suffisamment modulable pour remplacer cette étape de création des réseaux hydrographiques par d'autres techniques n'utilisant pas forcément des grammaires (subdivision de polygones [395], subdivisions sur grille [131]).

5.3.2.1 Sélection du nœud

Le nœud qui doit être étendu est sélectionné parmi une liste de nœuds-candidats (ou nœuds fertiles) en utilisant une heuristique qui prend en compte à la fois l'élévation et l'indice de priorité. En combinant ces deux critères, on permet la création simultanée de plusieurs réseaux de drainage hiérarchiques qui se partageront l'espace disponible. De plus, la possibilité de modifier l'importance de l'un des deux critères (l'élévation ou la priorité) donne à l'utilisateur un contrôle sur la forme des réseaux créés. Si l'indice de priorité est favorisé, l'algorithme va favoriser certaines rivières dès le début du processus. À l'inverse si on sélectionne les nœuds dont l'élévation est minimale, l'algorithme va générer des bassins de drainage en priorité dans les plaines.

Rappelons que $\mathbf{X} \subset \mathbf{R}$ symbolise l'ensemble des nœuds-candidats créés dans le graphe **G** au cours d'une étape de colonisation. La méthode suit ces trois étapes (Fig. 95) :

1. on cherche l'élévation minimale z parmi tous les candidats

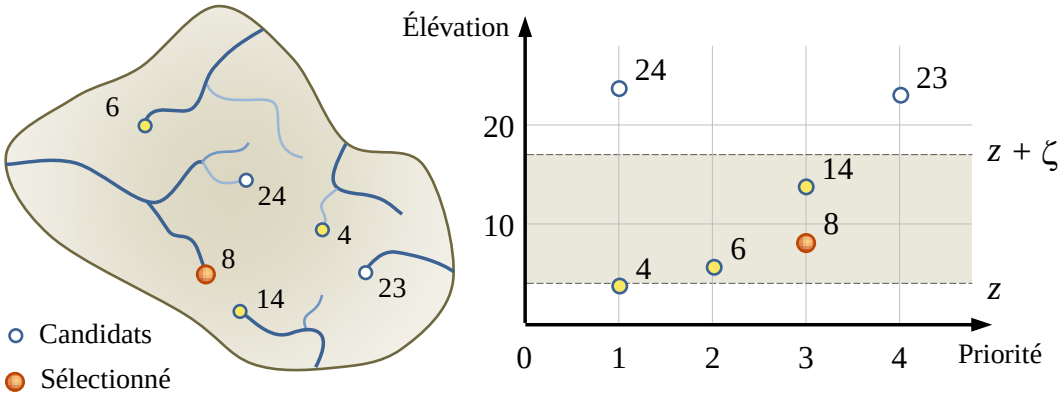


FIGURE 95 : L'algorithme de sélection d'un nœud est paramétré par une fonction qui combine l'élévation de ce nœud et sa priorité. L'algorithme sélectionne d'abord l'ensemble des nœuds-candidats dans un intervalle d'altitude $[z, z+\zeta]$ (z étant l'altitude minimale parmi l'ensemble de tous les candidats) puis la procédure filtre uniquement les nœuds dont la priorité est la plus élevée et enfin, parmi ces derniers, renvoie celui dont l'altitude est la plus basse.

2. on considère le sous-ensemble $\mathbf{X}_\zeta \subset \mathbf{X}$ de tous les nœuds admissibles dont les élévations sont suffisamment proches $[z, z+\zeta]$;
3. on choisit dans ce sous-ensemble de nœuds-candidats $\mathbf{X}_\zeta$ le nœud $R_\mathbf{X}$ qui a la priorité la plus importante. Si plusieurs nœuds respectent ce critère, on sélectionne celui d'altitude la plus basse.

Le paramètre $\zeta \in [0; +\infty[$ contrôle la longueur du réseau hydrographique en limitant la portée de l'élévation entre deux nœuds $\mathbf{X}_\zeta$. Quand $\zeta \approx 0$, l'algorithme correspond à une sélection du nœud le plus bas et tend à combler les zones plates avec de nombreuses ramifications. Quand ζ augmente, le nœud sélectionné est le candidat avec la priorité la plus grande, construisant ainsi des réseaux hydrologiques fortement structurés autour de quelques fleuves. Ces deux cas limites de la stratégie de sélection – qui ne sont pas utilisés en pratique – sont illustrés Fig. 96.

Il peut également être intéressant de faire évoluer le paramètre ζ au cours du temps pour favoriser la création de certains bassins-versants très structurants au début de l'algorithme. L'attribution des priorités à l'initialisation des embouchures (au moment du *seeding*) peut également être contrainte afin d'attribuer les priorités les plus élevées aux nœuds se trouvant dans les parties concaves du contour. De plus, des tests supplémentaires peuvent interdire à deux rivières de hautes priorités d'être trop proches l'une de l'autre.

5.3.2.2 Choix de l'expansion

Les nœuds du graphe sont construits en appliquant les règles décrites dans le Tableau 4. Nous utilisons un processus de réécriture similaire à une grammaire qui fait intervenir deux symboles non-terminaux notés α et β. Ces non-terminaux représentent respectivement les nœuds-candidats et les nœuds instanciés. Le seul symbole terminal, noté τ, représente un nœud qui a été ajouté au graphe et qui ne peut plus être étendu.

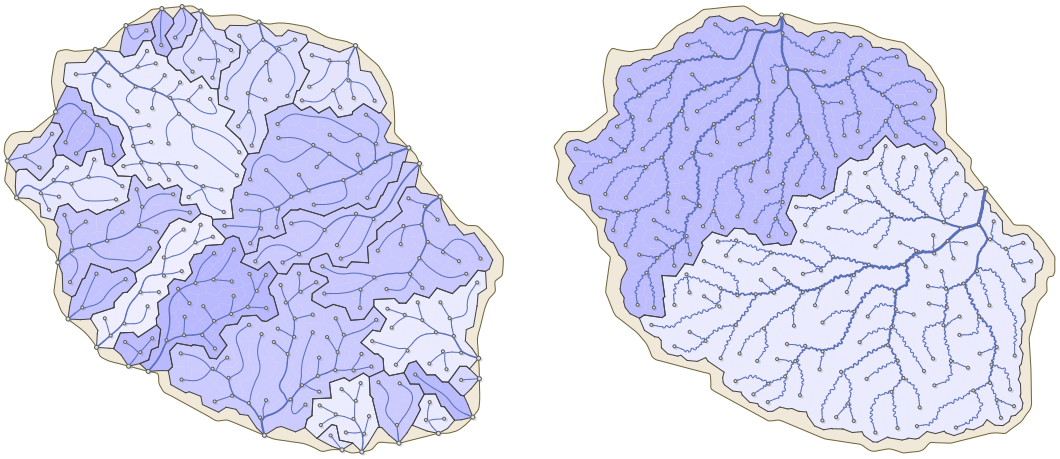


FIGURE 96 : Deux réseaux de drainage construits en suivant les cas-limites de l'algorithme de sélection. Quand $\zeta = 0$, des bassins-versants de taille comparable apparaissent sur le domaine. A l'inverse, quand $\zeta \to \infty$, seuls deux bassins-versants prioritaires sont construits.

Indice de priorité et nombre de Horton-Strahler

Les symboles non-terminaux dans le Tableau 4 sont paramétrés par un entier représentant l'indice de priorité s. Cela correspond au nombre de Horton-Strahler[165] souhaité, qui est une métrique de la complexité des embranchements d'un graphe géométrique représentant un réseau hydrographique. Cet algorithme est présenté en détail en Section 5.1.2.

On rappelle que le nombre d'Horton-Strahler d'une feuille est toujours $s = 1$ et que le nombre d'Horton-Strahler s d'un nœud est égal au maximum de ses enfants k sauf dans le cas où au moins deux enfants sont mesurés à k, alors $s = k + 1$.

On retrouve le principe du nombre de Horton-Strahler dans les règles (2.2) et (2.3) qui encodent le principe de branchement symétrique et asymétrique. Comme l'indice de priorité décroît potentiellement à chaque embranchement, il se peut, à une étape, que tous les nœuds-candidats restants aient un indice de priorité égal à un. Quand ce cas se produit, la grammaire utilise la première règle de production pour que le graphe puisse continuer à s'étendre. Cette règle ne correspond pas à l'algorithme d'origine de Horton-Strahler mais permet de recouvrir le domaine Ω. D'autres stratégies sont possibles : il est possible par exemple d'augmenter les indices de priorité de tous les nœuds de façon aléatoire ou en fonction de zones géographiques afin de re-privilégier le développement de certaines régions.

À la fin de l'algorithme, les indices de priorité du réseau hydrographique construit ne correspondent pas forcément au calcul des nombres d'Horton-Strahler (à cause de la règle de remplissage ou si un fleuve d'indice élevé voit sa croissance bloquée). On procède alors à un calcul exact de l'algorithme qui servira dans la suite de la génération (*e. g.* au moment de la création des confluences).

Tableau 4 : Grammaire permettant de créer des bassins-versants. Ces règles sont inspirées de la notion de jonctions symétriques et jonctions asymétriques décrites par l'algorithme de Horton-Strahler.

N°			Règle de production	Action supplémentaire
0.		$\xrightarrow{\text{axiome}}$	$\alpha_1(s_1)\ldots\alpha_n(s_n)$	
1.1	$\alpha(n)$	$\xrightarrow{\mathcal{P}_c}$	$\tau(n)\,\beta(n)$	Placer nouveau nœud
1.2		$\xrightarrow{\mathcal{P}_a}$	$\tau(n)\,\beta(n)\,\beta(m)$ où $m < n$	Placer nouveaux nœuds
1.3		$\xrightarrow{\mathcal{P}_s}$	$\tau(n)\,\beta(n-1)\,\beta(n-1)$	Placer nouveaux nœuds
2.	$\alpha(1)$	$\longrightarrow$	$\tau(1)\,\beta^p(1)$ avec $p \in [1;5]$	Placer nouveau(x) nœud(s)
3.1	$\beta(n)$	$\xrightarrow{\text{si valide}}$	$\alpha(n)$	$\mathbf{X} \leftarrow (\mathbf{X} \setminus \{R_\mathbf{X}\}) \cup \mathbf{N}$
3.2		$\xrightarrow{\text{sinon}}$	ε	$\mathbf{X} \leftarrow \mathbf{X} \setminus \{R_\mathbf{X}\}$

Application des règles

Les probabilités d'application des règles $\mathcal{P}_c$, $\mathcal{P}_a$, $\mathcal{P}_s$, avec $\mathcal{P}_c + \mathcal{P}_a + \mathcal{P}_s = 1$ sont définies par l'utilisateur et influencent le nombre relatif de branchements qu'il y aura dans le graphe final. Les nombres $\mathcal{P}_a$ et $\mathcal{P}_s$ font référence à la probabilité d'occurrence d'un branchement de type asymétrique (respectivement symétrique) alors que $\mathcal{P}_c$ correspond à la probabilité d'une simple extension sans branchement d'une rivière.

La règle 0 correspond à l'axiome de base. Les règles 1.2 et 1.3 s'inspirent de l'algorithme de Horton-Strahler et permettent de créer des jonctions respectivement asymétriques et symétriques. La règle 1.1 correspond à une extension simple sans confluence. Enfin la règle 2 permet – dans le cas où l'algorithme de génération ne ré-attribue pas de nouveaux indices de priorité – de remplir le domaine avec des jonctions multiples (qui correspondent au dernier niveau de subdivision). Les règles 3.1 et 3.2, bien qu'incluses dans la grammaire, testent la validité géométrique des nouveaux nœuds et rendent le nœud sélectionné à l'extension stérile (qu'il ait ou non développé une extension).

La colonisation s'arrête quand il n'est plus possible d'ajouter de nouveaux nœuds. Les règles d'expansion de la grammaire permettent d'indiquer quels sont les cours d'eau qui devraient s'étendre ; c'est l'étape suivante de création des nœuds et de validation géométrique de l'extension qui conditionne réellement la colonisation du domaine. Si un nœud sélectionné (grâce à son indice de priorité élevé ou par son altitude suffisamment basse) ne peut s'étendre à cause de contraintes géométriques (*i. e.* trop peu d'espace) alors, il est rendu stérile et ne pourra plus s'étendre. Des alternatives algorithmique existent : par exemple, le nœud empêchant l'extension peut être « capturé » (il change de parent topologique).

5.3.2.3 Expansion géométrique

Si un nouveau nœud doit être ajouté (une des règles 1.1-1.3 est utilisée), sa compatibilité avec le graphe géométrique déjà construit doit être vérifiée (cela correspond aux règles 3.1-3.2). Les nœuds invalides sont enlevés de la grammaire en appliquant l'epsilon production (règle 3.2).

Soit $R_{\mathbf{X}} \in \mathbf{R}$ le nœud sélectionné à l'extension. Soit $N = (\mathbf{p}, s, \rho, \phi)$ un des nœuds que nous voulons rajouter au graphe et E la nouvelle arête $(N, R_{\mathbf{X}})$. La longueur d'une arête géométrique est définie par la constante e. On note $R_i = (\mathbf{p}_i, s_i, \rho_i, \phi_i)$ les différents nœuds de l'ensemble $\mathbf{R}$.

Placement

Une fois que l'extension a été choisie (par exemple, la règle 2.1), on va procéder à la création de un ou plusieurs nouveaux nœuds (également appelés nœuds virtuels). Toutes les arêtes géométriques créées étant de longueur e (constante choisie par l'utilisateur de l'algorithme), on sait que le nouveau nœud N sera placé sur un cercle de centre $R_{\mathbf{X}}$ et de rayon e.

Tout nœud déjà créé du graphe $\mathbf{G}$ possède un vecteur de direction privilégié qui influence son extension. Ce vecteur v est généralement choisi en combinant deux données : une direction privilégiée tirée aléatoirement (ou définie par un champ de vecteurs) et une direction dépendante des ancêtres du nœud (qui pousse une rivière à continuer de s'étendre dans la même direction). Bien que le vecteur v indique une direction d'extension, nous utilisons également une distribution normale (centrée en 0, avec un écart défini pour donner une valeur d'angle comprise dans l'intervalle $[-330^\circ ; +330^\circ]$) pour calculer des déviations.

Validité du placement

La position $\mathbf{p}$ doit être à l'intérieur du domaine et suffisamment loin du contour Γ (Fig. 97 gauche), c'est à dire au moins à η (nous utilisons par défaut $\eta = 3/4 \times e = 1500[\mathrm{m}]$).

$$\mathbf{p} \in \Omega \quad \wedge \quad d(\mathbf{p}, \Gamma) > \eta.$$

Nous calculons la distance entre l'arête E et les autres arêtes du graphe pour éviter toute collision entre la nouvelle arête et le graphe déjà construit (Fig. 97 droite) : cette distance doit être supérieure à un paramètre σ défini par l'utilisateur (on utilise généralement $\sigma = 3/4 \times e = 1500[\mathrm{m}]$). La fonction $\sigma \cdot f(s)$ dépend du nombre de Horton-Strahler afin de maintenir deux fleuves importants suffisamment loin l'un de l'autre.

$$\forall E_i \in \mathbf{E} \quad : \quad d(E, E_i) > \sigma \cdot f(s).$$

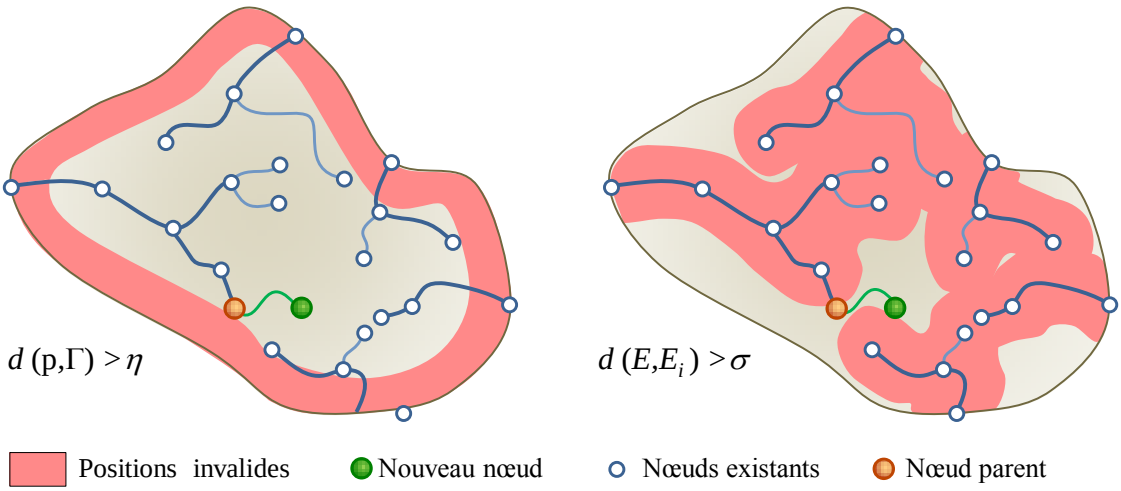


FIGURE 97 : Deux des conditions que nous testons lors de l'ajout d'un nœud : est-ce que le nœud est suffisamment loin du contour ? est-ce que le nœud n'est pas trop proche du graphe déjà construit ?

Validité des élévations

Durant la colonisation, les élévations des nouveaux nœuds doivent être strictement supérieures à leurs ancêtres afin de garantir un écoulement consistant depuis les sources jusqu'aux embouchures. Cette élévation est calculée à l'aide d'un coefficient de pente (non lié à la direction) qui est soit généré procéduralement, soit donné par l'utilisateur.

Dans les deux cas, cette pente des rivières (qui n'est qu'une valeur scalaire) définit uniquement la variation d'altitude et ne guide pas la direction de la colonisation. Projeté sur l'ensemble du terrain, cette **carte des pentes fluviales** permet de décrire intuitivement comment le réseau hydrographique va s'étendre (Fig. 98).

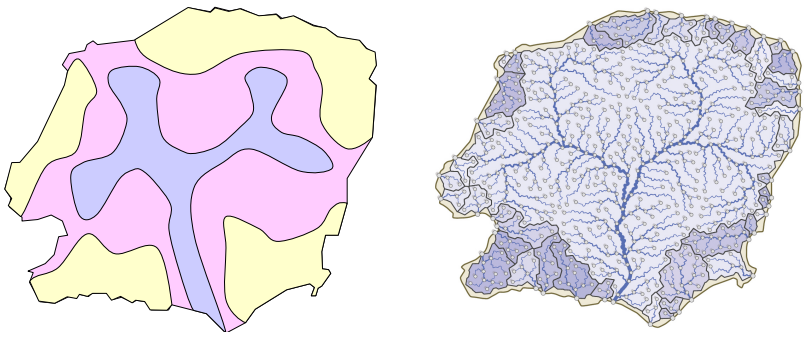


FIGURE 98 : La carte des pentes fluviales permet de contrôler intuitivement la forme que va prendre le réseau hydrographique.

Dans le cas où l'élévation du réseau hydrographique est calculée alors même qu'il est construit, l'élévation de $\mathbf{p}$ doit également être compatible avec l'élévation des autres nœuds du graphe. De plus $\mathbf{p}$ doit être à une altitude supérieure par rapport à son ancêtre $R_{\mathbf{X}}$. Nous contraignons la génération du terrain afin que la pente maximale du graphe soit inférieure à une borne κ qui

dépend de la position du point. La constante κ représente une borne supérieur de la fonction de pente fluviale. Ainsi, nous pouvons être certains que le nouveau point $\mathbf{p}$ satisfait la condition de Lipchitz suivante :

$$\mathbf{p}_z - \mathbf{p}_{z_i} < \kappa(\mathbf{p}) \cdot d(\mathbf{p}, \mathbf{p}_i).$$

Cette contrainte permet d'éviter la création de falaises trop abruptes (Fig. 99).

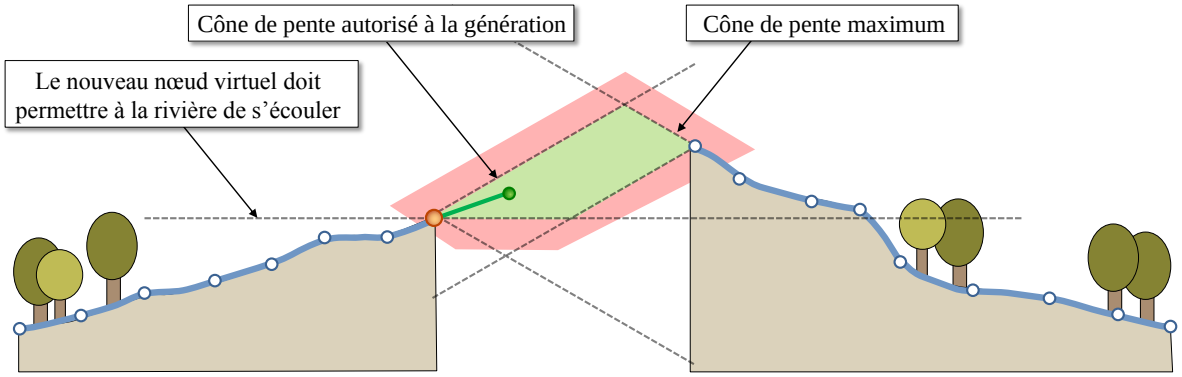


FIGURE 99 : Des tests d'élévation permettent de garantir un écoulement des rivières jusqu'aux embouchures et évitent la création de trop grands dénivelés.

Contrôle de la colonisation

Faire varier les paramètres de la génération procédurale de rivières permet d'obtenir des résultats aux effets différents comme il est montré sur les images Fig. 100. L'ensemble de paramètres $(\mathcal{P}_c = 0.2, \mathcal{P}_s = 0.7, \mathcal{P}_a = 0.1)$ produit des bassins-versants relativement déformés (image de gauche) avec peu de fleuves importants et beaucoup $(> 75\%)$ de petits cours d'eau ayant un nombre de Horton-Strahler égal à 1 (Fig. 100 gauche). À l'inverse, l'ensemble de paramètres $(\mathcal{P}_c = 0.2, \mathcal{P}_s = 0.1, \mathcal{P}_a = 0.7)$ produit des réseaux de drainage avec des bassins-versants de tailles équivalentes (Fig. 100 droite). Dans le second cas, les rivières importantes sont plus grandes car leur indice de priorité a été gardé plus longtemps durant la génération du réseau hydrographique. Ces bassins-versants sont donc structurés autour de ce grand fleuve.

Nous quantifions la différence entre ces deux graphes en évaluant le nombre d'arêtes pour chaque nombre de Horton-Strahler et en comparant leur distribution statistique. Si la probabilité $\mathcal{P}_s$ est élevée, l'indice de priorité aura tendance a décroître rapidement alors que si $\mathcal{P}_a$ est important, les fleuves importants vont grandir longtemps et auront une influence plus importante sur la forme des bassins. Si nous montrons que ces paramètres ont bien une influence sur la construction du réseau, ils sont moins intuitifs pour l'utilisateur et impactent globalement le système.

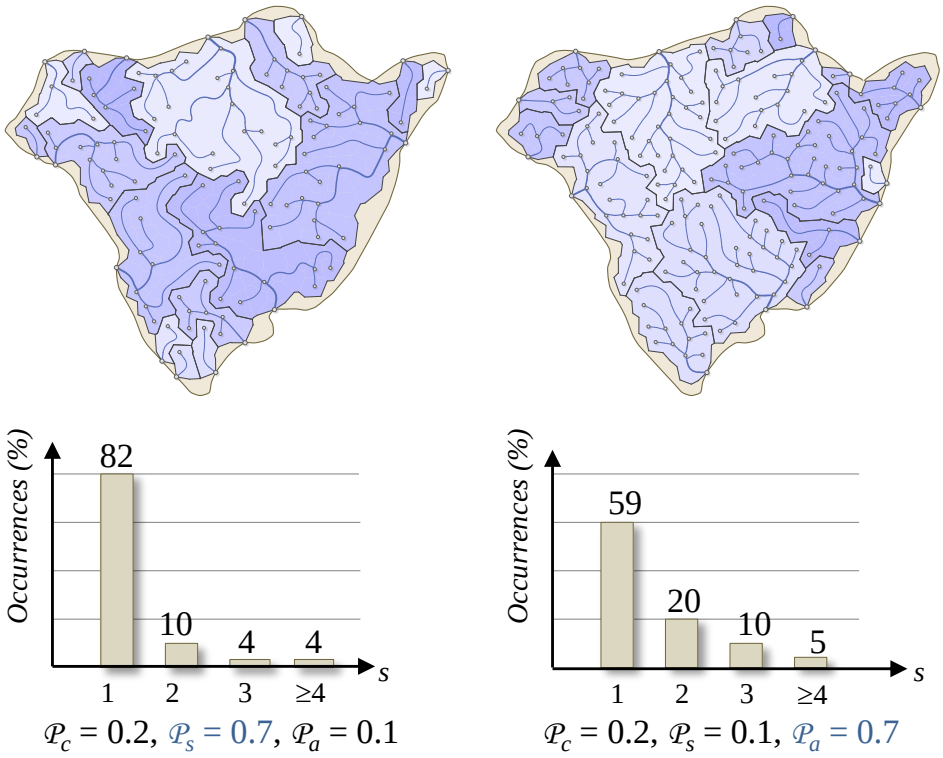


FIGURE 100 : Deux réseaux de drainage construits avec la même carte de pente et le même paramétrage de l'algorithme de sélection mais où les valeurs de probabilité d'embranchement sont différentes. On remarque que la probabilité de jonctions symétriques fait rapidement baisser la priorité de nombreux cours d'eau.

5.4 Génération des informations sémantiques

Maintenant que le réseau hydrographique a été créé, nous cherchons à générer un maximum d'informations sémantiques qui décrivent le relief. Certaines de ces informations sont générées sur l'ensemble du domaine, d'autres à l'intérieur d'une cellule correspondant à un morceau du domaine.

5.4.1 Génération des informations régionales

Il est possible de partitionner le domaine Ω en un ensemble de cellules. Cette segmentation va permettre de générer les lignes de crêtes du domaine. Elle structure également le terrain en un ensemble de bassins-versants. Enfin, les étapes restantes de l'algorithme (raffinement des trajectoires, construction des jonctions, génération du modèle de terrain) vont pouvoir être effectuées sur chaque cellule indépendamment : cela permet une parallélisation des opérations mais également un contrôle utilisateur (par exemple, en ne générant qu'une partie du domaine Ω).

5.4.1.1 Segmentation

Le réseau hydrographique est représenté par un graphe ; chaque nœud de ce graphe possède une position $\mathbf{p}_i$. Il est possible de segmenter le domaine en calculant un diagramme de Voronoï sur l'ensemble de ces positions. Le domaine Ω est alors décomposé en un ensemble de cellules $\mathbf{V} = \{V_i\}$ (Fig. 101 gauche).

Cette segmentation donne généralement des cellules de forme et de taille relativement homogènes (sauf les cellules « de mer »).

Rappelons que dans le graphe **G**, toutes les arêtes reliant des nœuds ont une même longueur e. Rappelons également qu'un nouveau nœud de rivière ne peut pas être placé trop près du réseau déjà construit (plus exactement $\forall E_i \in \mathbf{E} \ : \ d(E, E_i) > \sigma \cdot f(s)$).

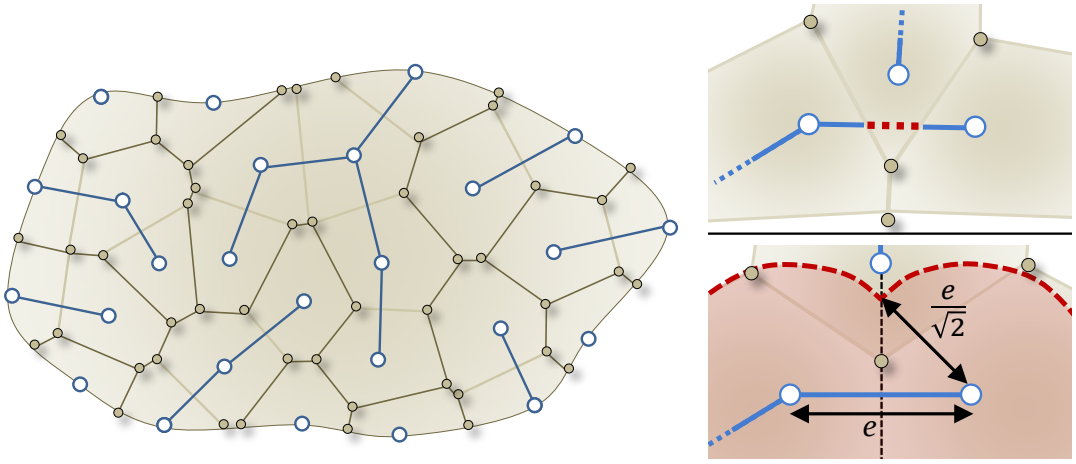


FIGURE 101 : (Gauche) La segmentation de Voronoï nous permet d'obtenir un ensemble de cellules convexes dont les bords approximeront les crêtes des montagnes.
(Droite) À partir du moment où le test de placement vérifie $\sigma > e/\sqrt{2}$, les cellules construites ont la propriété de n'intersecter aucun cours d'eau voisin.

On veut toujours que :

$$\sigma > \frac{e}{\sqrt{2}}$$

Cette propriété mathématique est nécessaire car elle garantit qu'une cellule de Voronoï qui est construite autour d'un nœud de rivière (souvent correspondant à une confluence) délimite une zone qui ne dépend que de ce cours d'eau (*i. e.* quand deux nœuds de rivières sont reliés par une arête, cette dernière ne traverse que les deux cellules correspondantes (Fig. 101 droite).

5.4.1.2 Regroupement des bassins-versants

On rappelle qu'un bassin-versant est défini comme l'ensemble du territoire sur lequel chaque goutte de pluie s'écoulera vers un unique exutoire.

Après avoir segmenté le domaine Ω en un ensemble de cellule de Voronoï V_i, il est possible de calculer un bassin-versant pour chaque exutoire $\mathbf{s}$ de chaque cellule. Le bassin-versant associé est défini comme l'ensemble des cellules connectées en aval $V_k \in \mathbf{V}$ à cette sortie d'eau (Fig. 102). En pratique, on traverse chaque cours d'eau depuis une source jusqu'à l'embouchure et on rajoute récursivement sur son passage l'union des cellules parcourues.

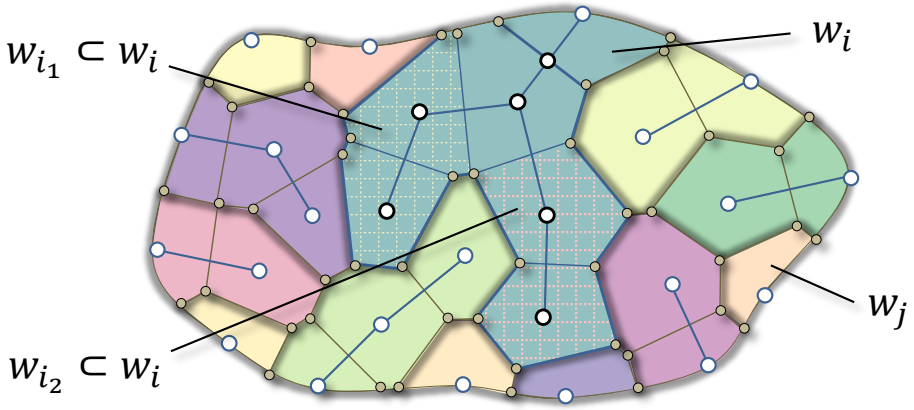


FIGURE 102 : Les bassins-versants sont construits pour chaque cellule en parcourant l'ensemble des cellules en amont du cours d'eau. On maintient une hiérarchie de sous-bassins-versants pour les cours d'eau comprenant des confluences.

Construire des bassins-versants nous permet d'estimer l'écoulement fluvial qui contribue à la définition de la géométrie des rivières. Le calcul exact du débit est un problème complexe qui dépend de nombreux paramètres comme le climat (pour déterminer l'intensité des précipitations ou de l'évaporation) ou la composition des sols (pour intégrer les calculs d'infiltration).

Nous utilisons un modèle plus simple basé sur une loi de puissance observée empiriquement en géomorphologie [104]. Définissons l'aire du bassin-versant comme $\mathcal{A}$ $[\mathrm{m}^2]$. Le débit moyen ϕ de la rivière $[\mathrm{m}^3\,\mathrm{s}^{-1}]$ est donné par la formule suivante :

$$\phi = 0.42 \cdot \mathcal{A}^{0.69}.$$

L'aire des bassins-versants $\mathcal{A}$ est approximée par la somme des aires des cellules connectées à la sortie d'eau $\mathbf{s}$. Il est alors possible de calculer le débit sortant ϕ de n'importe quelle cellule V (Fig. 103). Cette équation prend en compte l'évaporation et l'infiltration : c'est pourquoi la relation n'est pas linéaire et le volume total n'est pas préservé. Cette formule, utilisée en chaque exutoire, donne néanmoins une approximation plus réaliste du débit qu'en sommant les débits provenant des entrées d'eau en amont d'une cellule.

5.4.1.3 Génération des crêtes

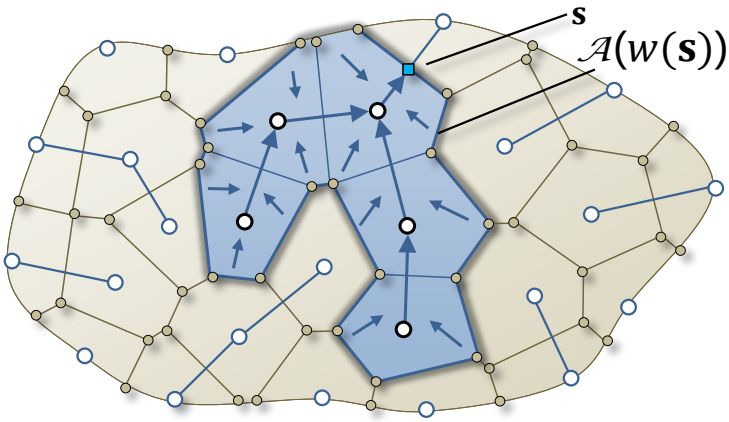


FIGURE 103 : À l'exutoire **s** correspond le bassin-versant $W_{\mathbf{s}}$ (en bleu) dont l'aire est notée $\mathcal{A}(W_{\mathbf{s}})$. Ce calcul du débit est effectué pour l'exutoire de chaque cellule grâce à une loi de puissance.

La segmentation du domaine Ω en un ensemble de cellules de Voronoï permet également de construire de manière automatique un réseau de crêtes dual. Ces crêtes vont permettre de délimiter les différents bassins-versants. Le graphe dual du réseau hydrographique produit un ensemble de segments (les arêtes délimitant chaque cellule polygonale) mais tous ne seront pas considérés comme des crêtes.

Pour chaque cellule de Voronoï, on distingue deux types d'arêtes : celles qui n'intersectent pas le graphe hydrographique définissent des crêtes, et les autres (Fig. 104). Sur ces dernières se trouve soit une entrée fluviale (*i. e.* un point de rivière en amont) notée $\mathbf{e}_k$, soit une sortie (*i. e.* un point de rivière en aval) correspondant à un exutoire, notée **s**.

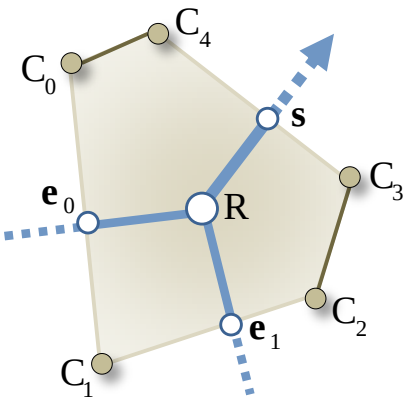


FIGURE 104 : Notations dépendant d'une cellule.

Le calcul de l'élévation des crêtes est primordial puisqu'il permet de garantir un écoulement cohérent. Chaque point de crête C est positionné à égale distance d des centres **a**, **b**, **c** qui sont les points de rivières du graphe. Cette crête C doit donc avoir une élévation supérieure à **a**, **b** et **c** afin que l'écoulement soit cohérent sur l'ensemble du domaine Ω.

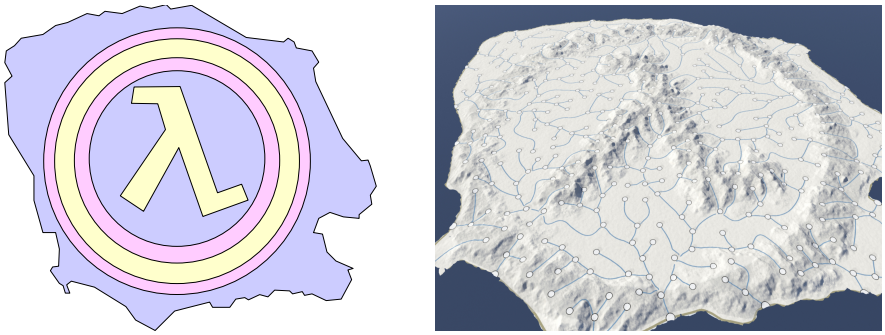


FIGURE 106 : La carte des pentes montagneuses permet de contrôler intuitivement les zones de montagnes.

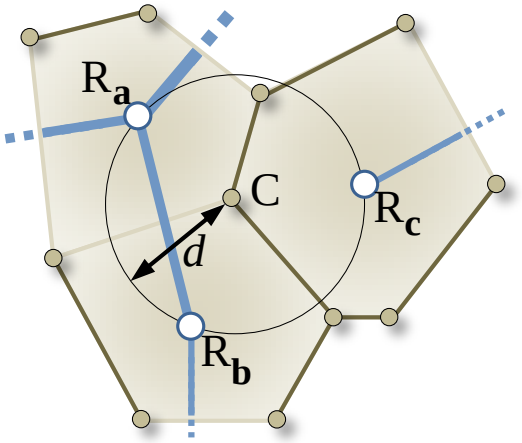


FIGURE 105 : Chaque point de crête C est équidistant (sur le plan Oxy) de 3 nœuds de rivières.

Nous calculons l'élévation C_z avec la formule suivante :

$$C_z = \max(\mathbf{a}_z, \mathbf{b}_z, \mathbf{c}_z) + \lambda(C) \cdot d$$

où $\lambda \in [0;0.25]$ est une autre fonction de pente qui décrit si le terrain est montagneux ou plat, et d est la distance entre la crête et le nœud de rivière le plus proche (Fig. 105). Cette carte, appelée cartes des pentes montagneuses (*slope map*), décrit quelles parties du terrain donneront des plaines, des plateaux, des vallées ou des montagnes (Fig. 106). En pratique, la fonction λ fait généralement intervenir soit une fonction de bruits, soit une carte de valeurs esquissée par l'utilisateur. De plus, cette fonction peut être modifiée en fonction de la distance à la côte ou de l'élévation des nœuds de rivières afin de générer des vallées plus douces ou des éléments plus abruptes, comme des falaises.

5.4.2 Génération des informations locales

Il est maintenant nécessaire de générer des informations plus précises dans chaque cellule de Voronoï. Les cours d'eau extraits du graphe représentant le réseau hydrographique ne sont matérialisés que par des segments droits. La création des bassins-versants a permis de connaître le débit d'eau à l'exutoire de chaque cellule. On a également pu calculer pendant la création du réseau hydrographique, un nouveau nombre de Horton-Strahler avec l'algorithme traditionnel utilisé en hydrologie.

On va maintenant décomposer les segments de rivières et leur attribuer un type. Notre méthode s'inspire de la classification de Rosgen [326] qui définit neuf catégories de rivières en fonction de leurs trajectoires et de leurs pentes. Chaque classe de rivière a un type de trajectoire

(**A+**, **A**, **B**, **C**, **D**, **DA**, **E**, **F**, ou **G**) et un profil décrivant le lit du cours d'eau. Cette classification inclut la composition géologique du lit (roches, pierres, gravier, sable, vase, argiles) dans les descripteurs utilisés.

L'attribution du type est effectuée à l'aide d'un algorithme *ad hoc* qui respecte certaines contraintes. Il est également possible de construire un ordre partiel des types de Rosgen. Nous assignons à chaque nœud de rivière une classe en nous basant sur la pente de la rivière et la distance à l'embouchure. Les rivières qui sont proches de la côte (en utilisant une distance géodésique le long de la courbe) sont étiquetées comme des rivières tressées (des rivières ayant plusieurs lits séparés par des bancs de sables et correspondant aux types **D** ou **DA**). De façon similaire, les embouchures de rivières avec un débit suffisamment important sont marquées comme des deltas, cas particulier des rivières à tresses.

Le type de chaque arête de rivière est déduit des types de ses deux sommets. Les transitions entre deux types sont déterminées à l'aide d'une table de correspondance.

Un cas particulier à gérer consiste à créer l'ensemble des confluences. Ces jonctions sont construites itérativement dans chaque cellule V. Chaque jonction est connectée à une jonction voisine. Dans une cellule de Voronoï, se trouve au plus une sortie fluviale (zéro dans le cas d'une cellule représentant une embouchure) et entre 0 et n entrées en amont.

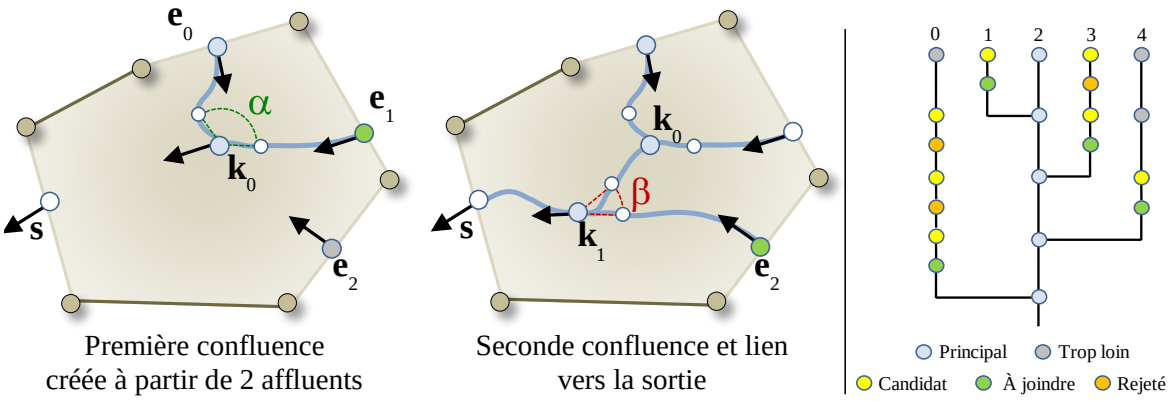


FIGURE 107 : (Gauche) Pour chaque cellule, on crée les confluences qui vont relier les entrées d'eau en amont. (Droite) L'algorithme fonctionne en reliant de manière incrémentale des cours d'eau voisins. Un exemple avec 4 confluences montre qu'il est nécessaire de faire un tri angulaire des entrées.

Le réseau hydrographique est créé en formant incrémentalement $n-1$ confluences à l'intérieur de chaque cellule V (chaque confluence est noté **k** sur la Fig. 107). Ce processus commence avec deux entrées d'eau en aval sur le contour de la cellule V. On note la première confluence $\mathbf{k}_0$. Chaque autre entrée en aval est connectée de manière incrémentale avec la dernière confluence créée. L'angle de connexion est calculé pour chaque jonction, en prenant en compte la position et le débit relatif des deux rivières. Quand la jonction fait intervenir deux cours d'eau ayant des débits très différents (donc ayant des nombres de Horton-Strahler différents), l'angle de jonction tend à être perpendiculaire . À l'inverse, une confluence entre deux rivières de même taille

correspondra à un faible angle de jonction. On parlera respectivement de jonctions en T et de jonctions en Y.

Une fois les jonctions construites, on raccorde les cours d'eau aux confluences et on raffine les trajectoires en fonction de leur type de Rosgen. Chaque courbe est décomposée récursivement en un ensemble de morceaux de cubiques de taille à peu près égale (cette longueur est définie par l'utilisateur et dépend également du type de Rosgen). Pour une plus grande facilité d'utilisation, on utilise une représentation par courbe de Hermite (2 points de contrôles avec 2 tangentes).

FIGURE 108 : Différentes trajectoires raffinées en fonction de leur type de Rosgen. Les courbes montrées sont de types **B** (bleu clair), **G** (bleu moyen) et **D** (bleu foncé).

Les points de contrôle sont perturbés en fonction du type de cours d'eau. En particulier, les types **A+**, **A**, **B** gardent une trajectoire relativement droite alors que les types **E** et **G** sont construites en créant itérativement des lacets ou méandres (Fig. 108). Pour cela, les trajectoires droites sont raffinées sans perturber les points de passage de la rivière (qui restent proches du trajet original) mais en faisant varier les tangentes pour qu'elles pointent dans la direction perpendiculaire (et ce de manière alternative gauche/droite).

La construction des méandres s'inspire des méthodes proposées par TEOH [375, 376] et de celles proposées par KUROWSKI [195, 196]. Les paramètres qui définissent la géométrie des cours d'eau dépendent également des débits au niveau de chaque arête.

5.5 Génération du modèle de terrain

Nous avons maintenant l'ensemble des informations nécessaires (graphe du réseau hydraulique, informations de débit ϕ, type de Rosgen des nœuds ρ, trajectoires précises) pour générer la géométrie du terrain. Le modèle de représentation de terrains continus par arbres de construction, présenté dans Chapitre 3, est utilisé pour représenter le relief.

L'arbre de représentation est construit en combinant des primitives de terrains (Section 5.5.1) et des primitives de rivières (Section 5.5.2). L'arbre regroupe un ensemble de petits morceaux de reliefs correspondant aux différentes cellules :

$$\mathcal{T}_V = \mathcal{B}(\{\mathcal{D}_i\}).$$

Les rivières sont intégrées par remplacement :

$$\mathcal{T} = \mathcal{R}(\mathcal{B}(\{\mathcal{T}_{V_i}\}), \mathcal{B}(\{\mathcal{R}_i\})).$$

Cet arbre peut être complété en rajoutant des primitives pour gérer plus finement les embouchures et les reliefs sous-marins.

5.5.1 Placement des primitives du terrain

Avant de générer la géométrie des rivières, il est nécessaire de construire l'ensemble des primitives de terrains qui doivent décrire le relief du domaine Ω. Pour ce faire, nous allons générer le relief de chaque cellule de Voronoï indépendamment en créant de petits arbres de construction. L'ensemble de ces cellules sera ensuite mélangé pour obtenir le terrain correspondant au domaine.

Pour créer le terrain nous avons besoin d'un ensemble de primitives $\mathcal{D}$ couvrant chaque cellule V et de leurs paramètres associés. Chaque primitive est principalement définie par une position 2D, une élévation, un rayon d'influence et un type de relief local (montagne, colline ou plaine).

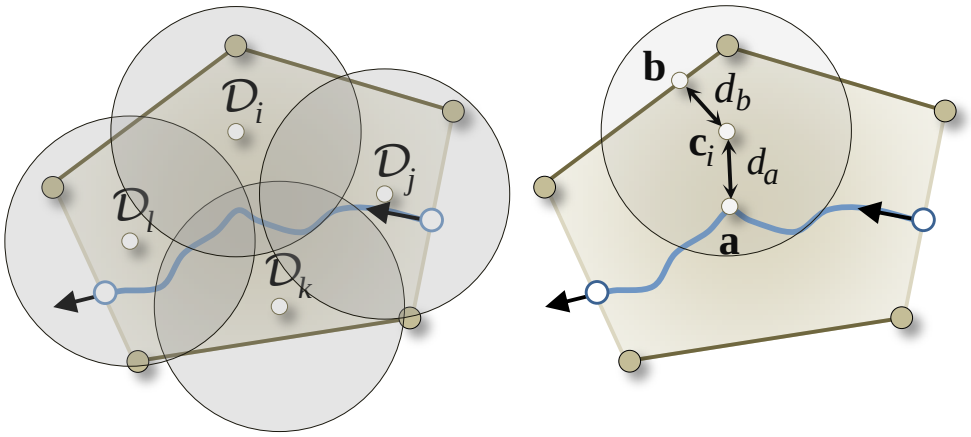


FIGURE 109 : (Gauche) La distribution des primitives de terrains doit couvrir le domaine de la cellule Ω_V. (Droite) L'altitude de chaque primitive dépend à la fois des élévations des crêtes et des rivières.

Les positions sont générées par un échantillonnage semi-stochastique en utilisant une distribution de Poisson [197] qui permet de construire des tuiles apériodiques (Fig. 109 gauche). Les rayons des disques sont augmentés itérativement de manière à ce que l'ensemble des primitives $\mathcal{D}$ couvre le domaine de la cellule Ω_V. Nous utilisons environ 50 échantillons par cellule. Utiliser moins d'échantillons (*i. e.* des disques de plus grands rayons) permet de calculer le terrain plus rapidement avec moins de coût-mémoire, mais produit des terrains plus grossiers.

Au cours de l'étape suivante, l'élévation des primitives (c'est-à-dire l'altitude au point **c**) est calculée comme une combinaison des élévations des crêtes et des rivières (que nous avons calculées à l'étape précédente) pondérées par leurs distances relatives à la primitive.

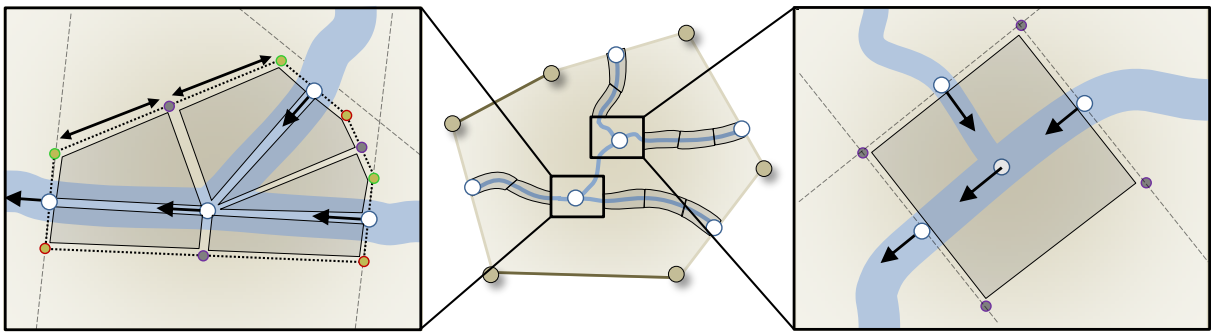


FIGURE 110 : La création de cours d'eau complexes (profils de rivières en tresses gérés par des cartes de hauteurs) nécessite parfois la décomposition des confluences en plusieurs sous-primitives.

En pratique, nous utilisons deux points caractéristiques : le projeté (noté **a**) de **c** sur l'ensemble des rivières et le projeté (noté **b**) de **c** sur les lignes de crêtes de la cellule de Voronoï (Fig. 109 droite). L'altitude de chaque primitive est donc calculée en fonction des altitudes des crêtes et des rivières qui sont elles-mêmes dérivées des cartes de pentes fluviales et des crêtes.

La forme du terrain peut être modulée par un bruit aléatoire. Les attributs du bruit (l'amplitude et la fréquence) qui sont associés à chaque primitive sont calculés en fonction de la distance à la rivière la plus proche d_a et de la différence d'élévation entre la rivière $\mathbf{a}_z$ et la crête $\mathbf{b}_z$. Modifier les attributs du bruit nous permet de produire des montagnes plus rugueuses ou, au contraire, des vallées plus douces. L'utilisation d'un bruit peut créer quelques minima locaux (qui devraient être considérés comme des bassins-versants) mais ils sont suffisamment petits pour ne pas modifier le contexte général d'hydrologie.

De plus, on déforme généralement la fonction d'interpolation des élévations (entre **c** et **a** ou **c** et **b**) pour que son profil transversal varie de région en région : un profil de type logarithmique sera adapté pour la création de vallées de collines, un profil linéaire sera adapté aux régions montagneuses et un profil de type exponentiel imitera les formes de vallées en U.

5.5.2 Placement des primitives de rivières

Les primitives de rivières sont générées avec des morceaux de courbes quadriques (Fig. 110 centre). Pour cela, on décompose les trajectoires de rivières raffinées qui ont été préalablement définies avec des courbes cubiques.

En plus des primitives de rivières analytiques présentées dans la Section 3.2, il est possible d'utiliser une bibliothèque de morceaux de rivières détaillés sous forme de cartes de hauteurs. Ces données ont été fabriquées par des outils d'infographie 2D/3D. Cette bibliothèque nous permet de placer des modèles très détaillés (par exemple des rivières en tresses). Les cartes de hauteurs sont projetées sur le terrain à l'aide de la paramétrisation en coordonnées (u,v) des morceaux de quadriques.

Chaque primitive de rivière est définie pour raccorder deux types de Rosgen (par exemple **A-A** ou **D-A**) afin de garantir une continuité géométrique. Pour les primitives-images, les raccords sont définis manuellement et grossièrement par des calques standardisés définissant le profil des rivières, l'opérateur de mélange masquant les défauts géométriques éventuels.

L'ensemble de ces primitives de rivières est paramétré par le débit d'eau (en $m^3 s^{-1}$). La relation entre le débit et la section transversale est donnée par la loi suivante :

$$\phi = V \cdot P$$

avec V la vitesse (en $m s^{-1}$) et S la section transversale (en m^2). Cette section s'approxime par la multiplication $l \cdot h$ où l représente la largeur et h la profondeur. Le type de Rosgen impose un ratio l/h. Il est donc possible, à partir d'un débit (et d'une vitesse moyenne estimée à partir de la pente) de retrouver les valeurs l et h.

Le cas des jonctions est particulier. On distingue deux types de confluences en fonction de leur nombre de Strahler ou de leur ratio de débit : les confluences pseudo-symétriques et les confluences asymétriques qui donnent deux types de jonctions géométriques : les jonctions en Y (Fig. 110 gauche) et les jonctions en T (Fig. 110 droite). Un moyen de réaliser des confluences en Y est d'avoir une primitive particulière construite à l'aide de 3 sous-primitives. Ces dernières correspondent à 3 quadrangles qui décomposent la jonction. Les jonctions en T sont construites à l'aide d'un unique quadrangle.

Toutes les primitives de rivières, provenant de l'ensemble des cellules du domaine, sont jointes par un opérateur de mélange.

5.6 Résultats

Nous avons implanté cette méthode dans une bibliothèque écrite en C++. Tous les terrains présents en illustrations dans le Chapitre 5 ont été générés avec notre algorithme. Le relief a été extrait d'un modèle de représentation sous forme d'arbre de construction. Les îles ont été générées à l'aide d'un ordinateur de bureau équipé d'un Intel® Core *i*7 avec une fréquence de 3 GHz et 16 Go de RAM. Les images photoréalistes avec de la végétation ont été produites en texturant et en décorant chaque relief dans e-on VUE xStream®. Les scènes complètes ont ensuite été rendues en les important dans MentalRay®.

5.6.1 Analyse qualitative

L'algorithme de génération procédurale permet de créer des îles de très grandes tailles. Les reliefs générés respectent les règles fondamentales en hydromorphologie (Fig. 111), *i. e.* toute goutte d'eau qui tombe en quelque partie que ce soit du domaine s'écoulera le long des pentes et finira par rejoindre la mer ou un lac interne.

En comparaison, les techniques de génération procédurales traditionnelles – qu'elles s'appuient sur la synthèse de bruit, sur des méthodes de subdivision, sur la modélisation fractale

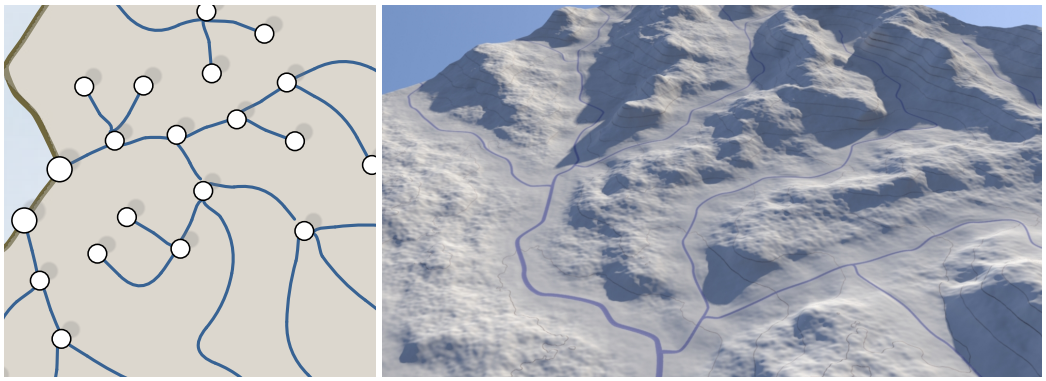


FIGURE 111 : Exemple d'un morceau de terrain généré par l'algorithme. Le relief est bien structuré autour de différentes vallées. L'écoulement est garanti sur l'ensemble du domaine.

ou encore sur le *faulting* – ne permettent pas de générer des réseaux hydrographiques structurant hiérarchiquement le terrain en vallées. Ces méthodes produisent généralement des terrains ressemblant à des reliefs artificiels ou extraterrestres (*i. e.* où se lisent peu de phénomènes d'érosion) ; on parle parfois de reliefs géologiquement jeunes (*fresh terrains*).

Les seules techniques arrivant à modéliser plus ou moins des reliefs visuellement plausibles sont les algorithmes de simulation d'érosion mais ces derniers sont lents et nécessitent beaucoup de temps de calculs. De plus, ces algorithmes sont souvent utilisés sur des terrains issus de méthodes de génération fractale : il en résulte des terrains de qualité moindre qui n'ont aucune structure de haut niveau (vallées, bassins-versants). En pratique, ces simulations génèrent souvent le même type de relief consistant en un ensemble de lacs entourés de chaînes de montagnes déstructurées mais à l'apparence érodée.

Notre approche est phénoménologique et ontogénétique : nous nous intéressons prioritairement à l'aspect visuel final du terrain, sans forcément simuler tous les effets physiques qui ont contribué à la sculpter. Cette approche tient compte de toute une série d'observations faites en hydrologie. L'algorithme présenté permet de générer des terrains qui ont une structure hiérarchique grâce à la création du réseau hydrographique avant la génération du relief. Ce réseau est généré grâce à une grammaire permettant une colonisation du domaine, s'inspirant de l'algo-

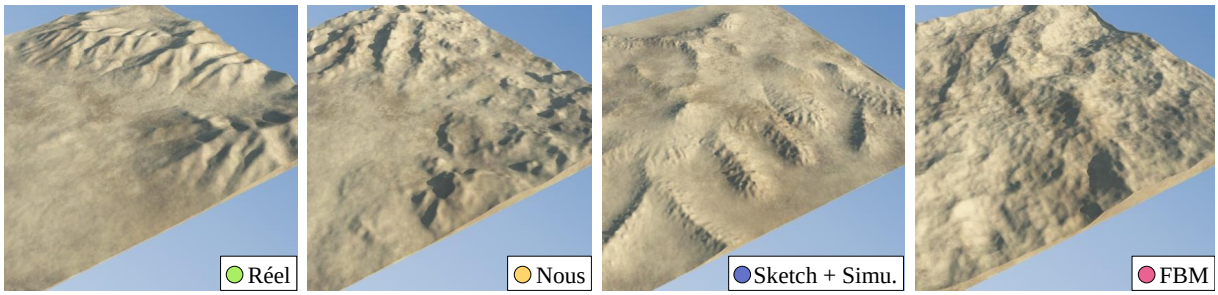


FIGURE 112 : Quatre morceaux de terrain issus de différentes méthodes : des données réelles d'élévation, un terrain produit par notre algorithme, un paysage créé en combinant esquisses grossières et simulation d'érosion hydrique, et enfin un relief généré par une fonction de bruit.

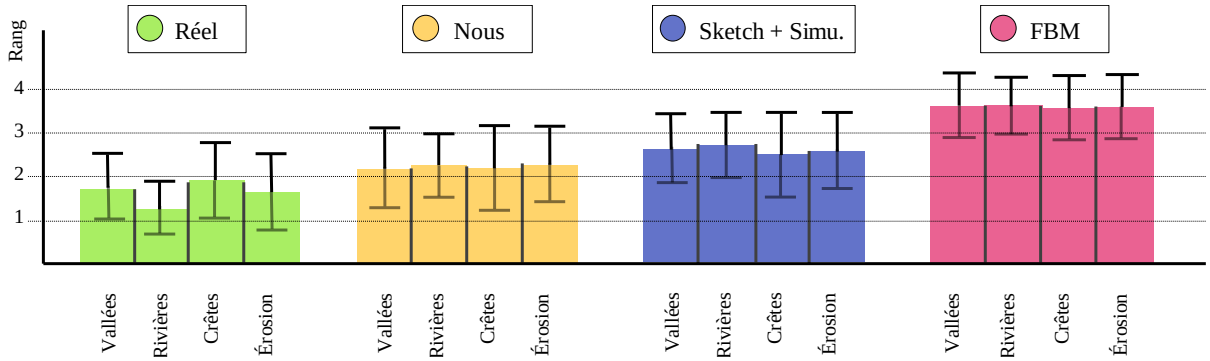


FIGURE 113 : La deuxième étude consiste à classer des terrains provenant de différentes méthodes selon 4 critères (réalisme des vallées, des rivières, des crêtes, de l'érosion). Le plus petit rang correspond à la méthode la mieux classée. Les moustaches indiquent l'écart-type à la moyenne des méthodes.

rithme de Horton-Strahler qui mesure la complexité d'un cours d'eau. Pour générer des cours d'eau avec une géométrie complexe, on s'appuie sur la classification de Rosgen, très utilisée en hydrologie et qui permet de décrire précisément l'apparence d'une rivière. Ainsi, à notre connaissance, cet algorithme est également un des rares capables de générer des cours d'eau avec une géométrie précise sans recourir à des simulations.

Études-utilisateurs

Nous avons réalisé deux études-utilisateurs où un échantillon homogène de 28 personnes (des étudiants compétents en informatique graphique ou en infographie) a pu comparer des terrains générés selon des méthodes différentes. Une comparaison visuelle de terrains issues de plusieurs approches en génération procédurale est proposée Fig. 112.

La première étude-utilisateurs consistait à noter (entre 0 et 5 selon que l'utilisateur est en accord ou en désaccord avec l'énoncé) les terrains selon 4 critères (« ce terrain est réaliste », « ce terrain est érodé », « il y a des vallées sur ce terrain », « il y a des rivières sur ce terrain »). Les résultats de cette première étude n'ont pas été significatifs car difficiles à interpréter en raison de plusieurs biais dans le protocole d'expérimentation.

Une deuxième étude-utilisateurs a été réalisée, où nous avons pu corriger les biais et où les utilisateurs doivent, cette fois-ci, classer des terrains les uns par rapports autres (afin d'obliger les utilisateurs a faire un choix).

Une étape consiste à demander à l'utilisateur d'observer 4 morceaux de terrain (10km $\times$ 10km) issus des 4 méthodes. Chaque terrain doit nécessairement être observé en plein-écran selon deux modalités : une fois avec un rendu non-texturé sans décoration (*i. e.* avec de l'*ambient occlusion*) et une fois avec un rendu texturé avec de la végétation (pour donner un repère d'échelle). Les rendus que voit chaque utilisateur correspondent à des prises de vues aériennes fixes. Une fois qu'un utilisateur a observé ces 4 terrains (*i. e.* 8 images en plein écran), il doit réaliser 4 classements distincts (« classer les 4 terrains selon le réalisme des vallées », « (...) selon le réalisme des rivières », « (...) selon le réalisme des crêtes », « (...) selon le réalisme de l'érosion »).

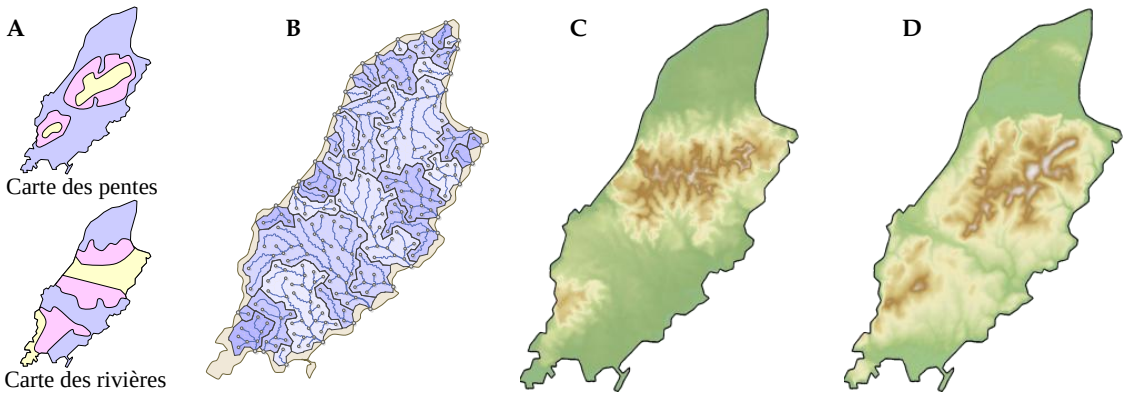


FIGURE 114 : (**A**) L'utilisateur esquisse les cartes de pentes qui vont contraindre la génération en une ou deux minutes. Le système complète le domaine en générant un réseau hydrographique (**B**) et un relief topographique cohérent (**C**). Si la méthode arrive à approximer la structure générale d'un relief réel (**D**), les détails, eux, ne sont pas rendues de manière fidèle.

Chaque utilisateur doit compléter 16 étapes : cela lui prend aux alentours de 20 minutes. À chaque étape, l'ordre d'apparition des types de terrains change (pour éviter les biais de classement). L'analyse de ces résultats semblent indiquer une corrélation significative [1] (Fig. 113).

Reproduction de la réalité

Enfin, un dernier moyen de valider l'algorithme est d'essayer de reproduire un terrain réel (Fig. 114). Après avoir approximé le contour de l'île de Man (une île située en mer d'Irlande), nous utilisons deux cartes de contrôles qui imitent grossièrement la forme du réseau hydrographique et des chaînes de montagnes de l'île. Le réseau et le terrain générés procéduralement confèrent à « notre » île un aspect général proche de celui de l'île de Man réelle, mais échouent pour ce qui est de la reproduction exacte des chaînes de montagnes. Une des limitations de notre algorithme vient du fait qu'il génère un réseau hydrographique à une seule et unique échelle, donnée par l'utilisateur (*i. e.* la taille des arêtes du graphe est constante). De plus, les profils des vallées sont difficiles à paramétrer ; or ce sont eux qui influencent l'aspect visuel dans les zones montagneuses. Le terrain a été généré en utilisant les paramètres par défaut de la grammaire, et à l'aide de deux cartes de contrôles. Ces dernières ont été esquissées manuellement, l'opération ne prenant que quelques minutes.

5.6.2 Contrôle

Le système de génération procédurale que nous proposons est facilement contrôlable. La relative simplicité de la méthode fait qu'un utilisateur non-initié comprendra rapidement le processus de génération. Des outils de contrôles sont disponibles à chaque étape. Tout d'abord,

1. La conception du protocole et l'analyse préliminaire des résultats ont été réalisés avec Manuel VEIT (HOLO3), Antonio CAPOBIANCO et Jérôme GROSJEAN (laboratoire ICUBE, équipe IGG) à Strasbourg.

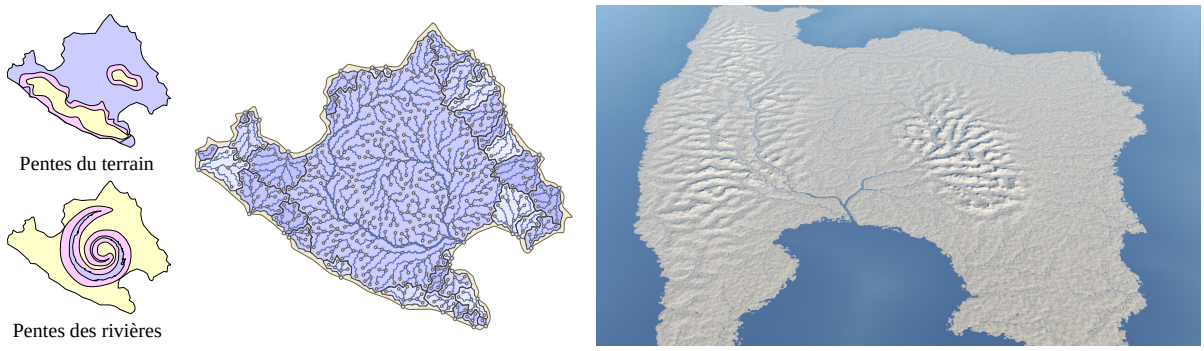


FIGURE 115 : Il est possible de contrôler intuitivement la création du réseau hydrographique. La spirale décrite dans les contraintes se retrouve bien dans le réseau final.

l'utilisateur choisit le domaine Ω en dessinant un ou plusieurs contours polygonaux. Il peut aussi esquisser manuellement une partie du réseau hydrographique (la position des embouchures, certaines rivières, la priorité de certains fleuves) qui se retrouvera dans l'île générée par l'algorithme.

L'utilisateur bénéficie également d'un contrôle de plus haut niveau. Il peut modifier les paramètres de génération (de la grammaire ou des algorithmes *ad hoc* de raffinements de trajectoires) ce qui a un impact sur les trajectoires de rivières construites. L'algorithme s'appuie sur deux cartes de contrôles : **la carte des pentes des rivières** et **la carte des pentes des crêtes** qui contrôle de façon grossière mais intuitive la génération du relief en indiquant où sont les zones montagneuses et où sont les plaines. Aussi complexes que soient les contraintes, la grammaire construit toujours un réseau hydrographique compatible et le terrain résultant respecte bien la contrainte physique d'écoulement des eaux (Fig. 115 et Fig. 116).

Notre système est également modulable. Chaque composant de l'algorithme peut être remplacé. En particulier, la phase de création du réseau hydrographique peut être remplacée par un algorithme de subdivision de polygones [395] ou de grilles régulières [131]. L'extension du modèle de représentation par arbres de construction impacte également la création du relief en proposant de nouveaux modelés (crêtes linéaires, fonctions de pentes érodées localement, ...).

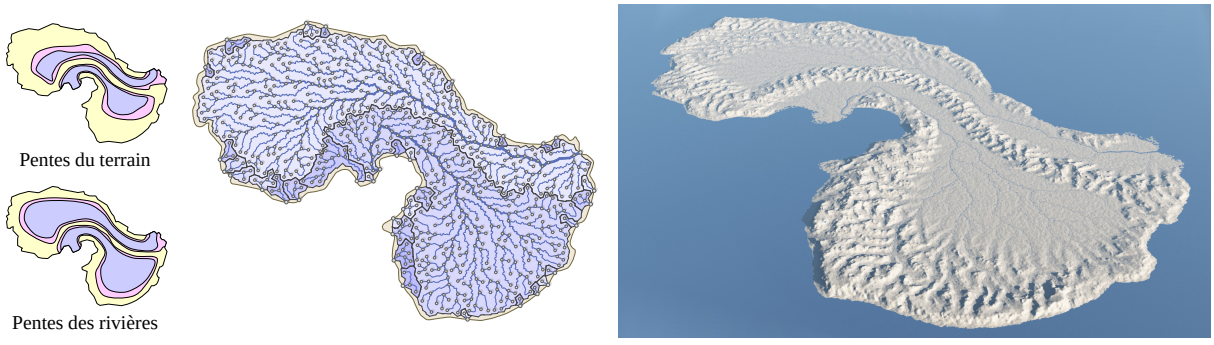


FIGURE 116 : Même dans des cas où les contraintes sont très complexes, l'algorithme réussit à construire un réseau hydrographique valide et un relief cohérent.

Tableau 5 : L'algorithme est capable de générer de grands terrains en quelques secondes. L'utilisation de cartes de contraintes complexes ralentit beaucoup la construction du réseau hydrographique.

Scène	Superficie	Hydro.	⏱ Étapes (s)			⏱ Total (s)
	(km^2)	(km)	Réseau	Infos. sup.	Arbre	
Fig. 111	31×31	2106	5.3	1.1	4.9	11.3
Fig. 114	50×50	399	0.1	0.1	1.8	2.0
Fig. 115	55×55	1787	4.6	0.7	3.8	9.1
Fig. 116	56×56	1978	4.0	0.9	4.7	9.6
Fig. 117	40×40	1515	0.5	0.4	4.8	5.7
Fig. 118	58×58	973	0.2	0.2	4.2	4.6
Fig. 119	55×55	1686	0.6	0.4	4.6	5.6

5.6.3 Performances

L'algorithme de génération est relativement rapide (de l'ordre de quelques secondes) et permet de construire des îles de très grande taille (leur superficie peut atteindre plusieurs centaines de kilomètres carrés) et avec une structure hydrologique réaliste. La méthode ne repose sur aucun calcul d'érosion, calculs souvent coûteux en temps. Le terrain généré est représenté sous la forme d'un arbre de construction (le modèle de représentation présenté Chapitre 3). C'est donc une description vectorielle peu gourmande en mémoire : en effet, la description de la plupart des îles générées et présentées ne nécessite que quelques kiloctets de données alors que les îles peuvent faire plusieurs dizaines de kilomètres de diamètre (Fig. 117). Chaque arbre de construction peut être visualisé en temps-réel grâce aux techniques présentées Chapitre 4.

L'algorithme s'articulant en 3 grandes étapes, nous pouvons restreindre la génération du terrain pour ne générer qu'une région en fonction de la position de la caméra ou des ressources de l'ordinateur. En effet, si le réseau hydrographique est construit sur l'ensemble du domaine, il est possible de ne raffiner qu'une partie des cellules de Voronoï. De la même manière, le placement de primitives de terrains et de rivières peut être effectué sur un sous-domaine sans impacter la qualité finale du terrain aux alentours de la caméra. On peut noter que l'île Fig. 116 a une superficie de 3387 km^2 et qu'elle est composée de 35188 primitives occupant en mémoire environ 4180 ko. La description vectorielle d'une partie du terrain correspondant à 30 km^2 ne représente que 1068 primitives et n'occupe que 125 ko.

Chaque île est représentée par des dizaines de milliers de primitives. Pour gérer des arbres de construction de cette taille, il est nécessaire de recourir à des opérateurs n-aires pour limiter la profondeur de l'arbre et la taille mémoire. Pour évaluer l'arbre et obtenir un maillage de

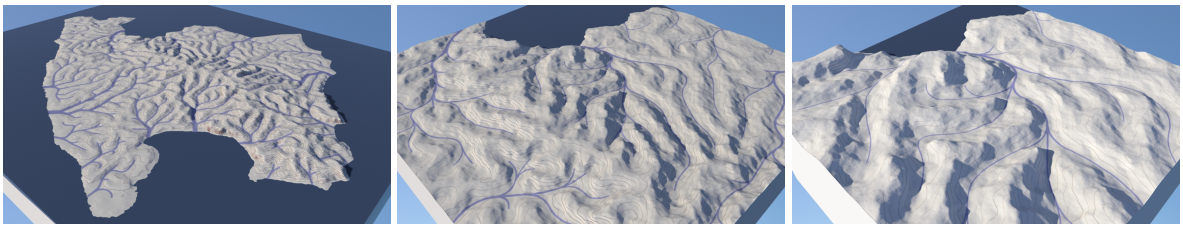


FIGURE 117 : Les îles générées ont un niveau important de détails et la structure du réseau hydrographique a un impact sur l'ensemble du domaine.

façon efficace (Tableau 5), l'utilisation seule des boites englobantes n'est pas suffisante et il est nécessaire de recourir à des structures accélératrices (*quadtrees* ou *loose grids*).

5.7 Limitations et perspectives

Nous présentons certaines limitations et perspectives de l'algorithme de génération procédurale de terrains.

5.7.1 Limitations

La principale limitation de notre approche est que l'algorithme a été **spécifiquement conçu pour générer des terrains structurés autour de l'érosion hydraulique à grande échelle**. Un terrain dans un climat sec peut être simulé si on ne creuse pas les rivières. Un terrain endoréique (qui est structuré autour d'un bassin-versant qui ne possède pas d'embouchure vers la mer et où l'ensemble des eaux de pluies ne peuvent quitter le bassin que par évaporation ou par infiltration) ne peut être construit qu'à partir d'un unique patch.

Une autre limitation vient du fait que les rivières se contentent de se rejoindre progressivement en aval. **Des phénomènes plus complexes** qui nécessitent que les rivières puissent se

FIGURE 118 : Si le réseau hydrographique influence le relief global du terrain, la géométrie des cours d'eau est également importante pour donner vie au paysage quand on l'observe depuis un point de vue proche du sol.

FIGURE 119 : Diffentes vues d'une rivière en tresse à plusieurs échelles. La végétation peut être directement corrélée aux placements de primitives de rivières.

diviser pour donner naissance à des deltas, à des tresses complexes, ou à des bras morts (*oxbow lakes*) **sont impossibles à gérer au niveau de la grammaire** et du graphe hydrographique. Il est toutefois envisageable de créer de tels cours d'eau en utilisant des primitives, analytiques ou à base d'images spécifiquement conçues, pour modéliser ces phénomènes (Fig. 118).

L'algorithme de construction de graphe de rivière et de propagation d'élévation peut engendrer des différentiels d'altitudes importants ce qui résulte en **des paysages d'aspect non-naturel**. La condition de Lipschitz sur le calcul des altitudes des nœuds de rivière réduit ce problème mais les transitions entre zones de plaines et zones montagneuses sont encore imparfaites.

En outre, le réseau hydrographique généré n'étant pas multi-résolution, il n'arrive pas à créer des montagnes de très grande taille dont les pentes seraient parcourues de petits cours d'eau (contrairement à des vallées proprement dites). Si les deux cartes de pentes (pour influencent les fleuves et les crêtes) sont fortement différentes, des fleuves relativement plats peuvent couper à travers des chaînes de montagnes. La méthode risque aussi de générer des paysages à l'aspect artificiel en faisant varier à l'extrême certains des paramètres de génération. L'équilibre entre le contrôle-utilisateur et la génération automatique devrait être étudié plus en détail.

5.7.2 Perspectives

L'algorithme étant modulable, chaque étape de la génération peut être amélioré ou remplacé. En particulier, **l'amélioration de la génération du réseau hydrographique nous paraît être la principale perspective de ces travaux**. Plusieurs étudiants de Master (en informatique) ont essayé, dans le cadre de leur travail, de créer des réseaux hydrographiques multi-échelles, que ce soit par un algorithme de subdivision sur des domaines polygonaux [395] ou sur des grilles régulières [131].

D'autres pistes sont envisageables, en particulier la génération du réseau sur des grilles hexagonales. Il n'est pas interdit de penser que le processus de colonisation pourrait s'appuyer sur d'autres mécanismes comme les RRT [208, 237, 238] (*Rapidly-exploring Random Trees*) ou la DLA [54, 410] (*Diffusion-limited aggregation*).

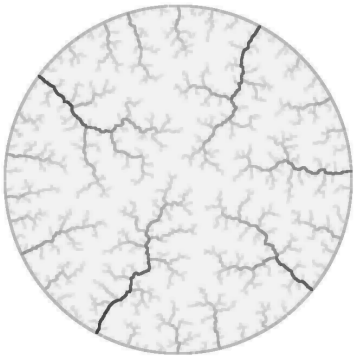


FIGURE 120 : Colonisation par *Rapidly-exploring Random Trees*.

L'amélioration de l'algorithme pourrait également **concerner le contrôle**. En particulier, la carte des pentes de rivières pourrait être complétée par un champ de vecteurs permettant de guider plus précisément la forme du réseau hydrographique. D'autres outils de contrôle capables d'améliorer la définition des profils de vallées et le placement de différents biomes permettraient de construire des paysages de façon plus intuitive.

Une autre piste à explorer serait d'inclure à l'algorithme **une génération procédurale de la végétation**, afin d'obtenir de manière automatique une description et une distribution des espèces végétales adaptées aux différents types de terrains, notamment le long des cours d'eau. Un prototype d'une version très simplifiée de ce système ainsi enrichi a déjà été utilisé pour plusieurs de ces images (Fig. 119 haut), la distribution des herbes, par exemple, étant liée à la distance aux rivières.

5.8 Conclusions

Nous avons proposé **un algorithme de génération procédurale de terrains capable de construire automatiquement des terrains dont le relief est structuré autour de vallées**. Un réseau hydrographique, créé à l'aide d'une grammaire, permet de garantir que toute goutte finira bien par s'écouler vers la mer. Mettant à profit les connaissances issues de différentes études du domaine de l'hydrologie (nombres de Horton-Strahler, classification de Rosgen), la modélisation des cours d'eau aboutit à des résultats au rendu réaliste (Fig. 119 et Fig. 121).

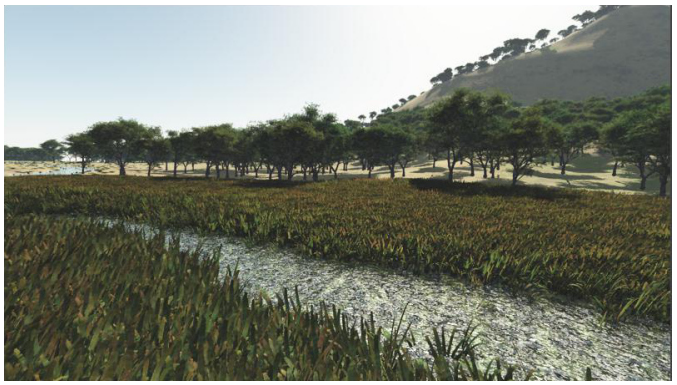


FIGURE 121 : La végétation permet de rendre les bords des rivières beaucoup plus riches visuellement.

Ce travail témoigne aussi de **la possibilité d'utiliser le modèle de terrains par arbres de construction** (présenté Chapitre 3) comme représentation intermédiaire (*i. e.* gérant la géométrie) dans le contexte d'un algorithme de génération procédurale. Le processus de création automatique de terrains que nous proposons se concentre sur la génération d'informations sémantiques et vectorielles (où se trouvent les rivières, quelles sont leurs trajectoires, *etc*) plutôt que

sur la géométrie. Cette dernière est gérée au travers du modèle de représentation des terrains par arbre de fonctions.

Comme le *pipeline* repose sur des étapes facilement compréhensibles par l'utilisateur, **il est aisé à reproduire**. Une belle preuve en est qu'un étudiant de Master de l'université de Wedel (Allemagne) a refait une implémentation Python [248] des étapes 1 et 2 de l'algorithme et a obtenu en quelques semaines des résultats proches des nôtres (Fig. 122 haut). La modularité du code a également permis d'expérimenter d'autres moyens de générer le réseau hydrographique, par exemple, avec une subdivision de polygones [395] (Fig. 122 bas).

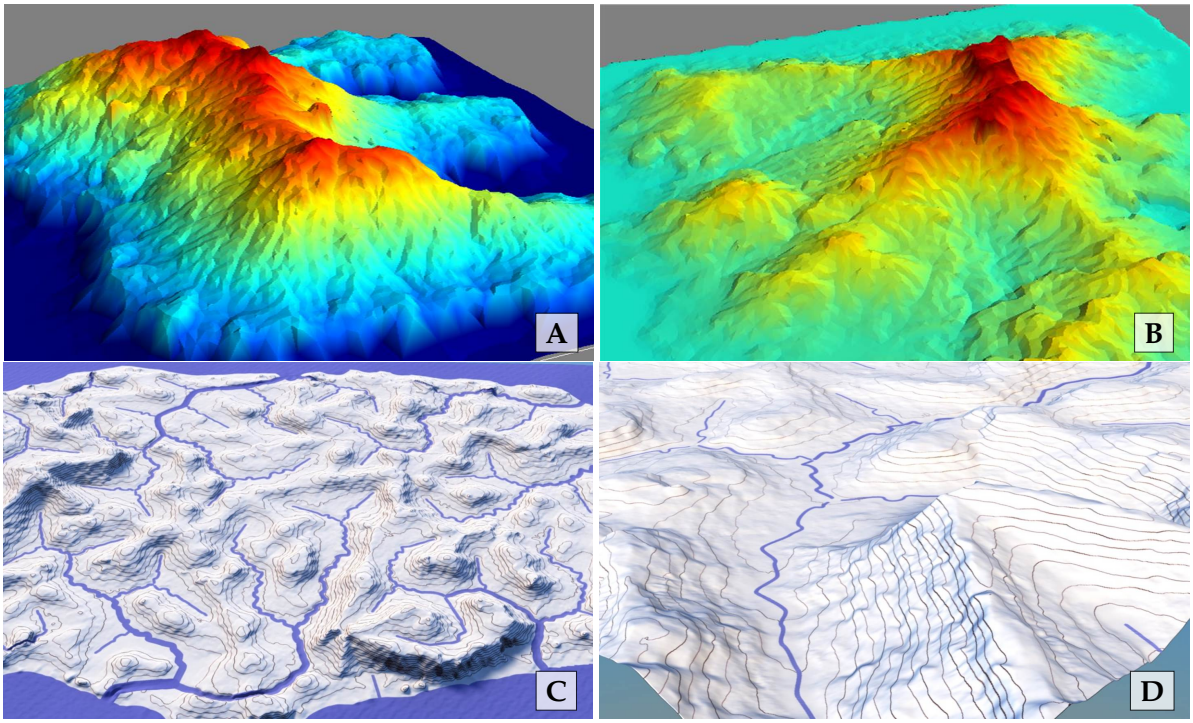


FIGURE 122 : (**A**,**B**) L'algorithme de génération est suffisamment intuitif pour être facilement reproduit. (**C**,**D**) Chaque étape de l'algorithme peut être remplacée ; ici, le réseau hydrographique est généré par une subdivision hiérarchique de bassins-versants polygonaux.

6

Conclusion

Now this is not the end.
It is not even the beginning of the end.
But it is, perhaps, the end of the beginning.

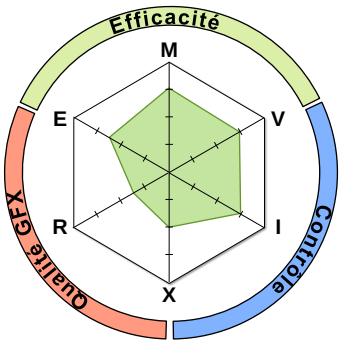
— Winston Churchill

Tout au long de nos travaux, nous avons exploré plusieurs aspects de la création numérique de scènes naturelles. Nous avons proposé **un modèle de données pour représenter des terrains continus** (Chapitre 3), **des algorithmes pour visualiser ce modèle** (Chapitre 4) et **un algorithme de génération procédurale** qui s'appuie sur ce formalisme (Chapitre 5). L'ensemble de ces travaux ont donné lieu à deux publications internationales [140, 141] ainsi que deux publications nationales [139, 142] ayant chacune remporté le 2ème prix du meilleur papier.

Analyse

Dans le Chapitre 2, nous avons proposé **un système de notation sous forme de diagramme radar**, qui permet d'évaluer des méthodes et des représentations selon 6 critères (**M**émoire, **V**itesse, **I**ntuitivité, e**X**pressivité, **R**éalisme, **É**chelles de taille et résolution) : **nous allons donc l'appliquer** au modèle de terrain par arbre de construction et à l'algorithme de génération de terrains.

Modèle de terrains par arbre de construction



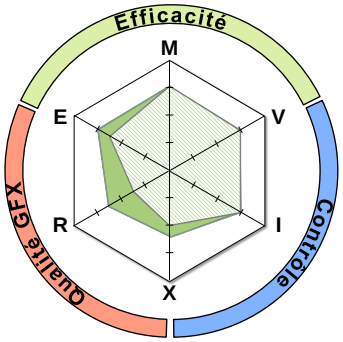
Analyse : l'objectif de nos travaux a été de proposer **une représentation à base de fonctions procédurales qui soit contrôlable**. L'utilisation d'un arbre de construction, similaire à la CSG ou aux BlobTrees, combiné à des primitives dont le support est compact, permet un contrôle (ou du moins une interprétation) relativement intuitive de ce que l'on construit.

L'utilisation de fonctions procédurales (*i. e.* construites à partir de fonctions de bruits) et vectorielles (*i. e.* des fonctions attachées à des objets géométriques comme des courbes ou des polygones) permet de construire le relief des terrains avec un niveau de détails important. Le modèle propose aussi de mélanger des primitives de plusieurs échelles et de plusieurs résolutions en même temps. La possibilité d'utiliser des données rasters (*i. e.* des cartes d'élévations) autorise les infographistes à importer des morceaux de reliefs très travaillés (par exemple, avec des algorithmes de simulation d'érosion).

Ce modèle est très efficace en terme de mémoire : chaque petite scène ne nécessite que quelques centaines de kilo-octets et les plus grands terrains, capables de modéliser plusieurs dizaines de kilomètres carrées, pèsent plusieurs mega-octets. Le modèle est relativement rapide à évaluer : il suffit de quelques secondes sur CPU pour extraire un maillage. Une implémentation GPU permet d'accélérer ce processus pour créer des maillages en temps interactif. L'on peut ainsi envisager la conception d'un outil temps-réel de création, de paramétrage et de placement de primitives. Notre prototype de *sphere tracer* amélioré a également montré que le modèle de représentation s'adapte à des algorithmes de rendu sans maillage.

Malheureusement, comme cette représentation est fonctionnellement *pure*, il n'est pas envisageable de faire des calculs qui nécessitent des voisinages. Il n'est donc pas possible d'implémenter des simulations d'érosion dans ce modèle. Seul le placement à la main et le paramétrage précis de chaque primitive permet d'obtenir des paysages réalistes. De même, aucun outil n'est proposé pour construire des méta-primitives afin de définir des paysages complexes et non-homogènes (par exemple, une zone de montagne possédant une structure interne, des lacs, *etc*).

Algorithme de génération de terrains



Analyse : la méthode de génération de terrains permet de compléter certaines des lacunes du modèle de base. Elle permet de **construire** avec une relative facilité **des paysages de très grandes tailles** (plusieurs dizaines de kilomètres de côté). La création au préalable d'un réseau hydrographique permet d'obtenir une structure du relief sous la forme d'un ensemble hiérarchique de vallées et de crêtes. Avec cet algorithme, on peut donc **obtenir un relief érodé, sans devoir recourir à des simulations** d'érosion coûteuses en temps de calculs.

Notre algorithme est également capable d'assembler de façon réalistes des primitives de rivières. C'est la première fois qu'un algorithme de génération essaye de modéliser des cours d'eau à très haut niveau de détails.

L'algorithme de génération procédurale nécessite quelques secondes de calculs pour générer un arbre de construction. Une fois cette arbre obtenu, il est envisageable de modifier localement l'arbre pour obtenir tel ou tel relief particulier.

Perspectives

Cette thèse ouvre de nombreux axes de recherche. Certaines de ces perspectives concernent le modèle de données, d'autres la visualisation, et d'autres encore l'algorithme de génération.

Extensions du modèle

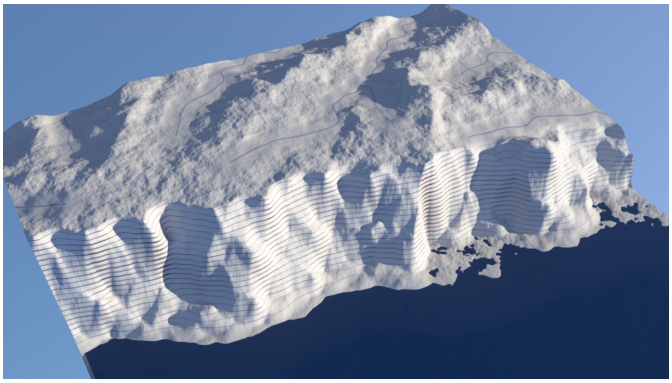


FIGURE 123 : Falaise visuellement convaincante générée par un *bug* dans le placement des primitives.

Les perspectives les plus prometteuses sont offerts par la possibilité d'extension du modèle. Actuellement, celui-ci se borne à la représentation d'un relief sous forme de multiples couches de matériaux superposées. Cela ne permet de construire que des terrains planaires (même si il est possible de générer des falaises visuellement plausibles comme montré dans la Fig. **??**). Une des premières extensions envisageables serait d'intégrer la gestion des surplombs, en combinant ce modèle avec le formalisme des surfaces implicites. Ceci engendre des problèmes qui touchent le contrôle (comment rendre intuitif l'utilisation des fonctions de poids en 3 dimensions ?) mais aussi sur l'efficacité de la représentation (il est alors nécessaire d'extraire une isosurface pour visualiser le terrain). D'autres

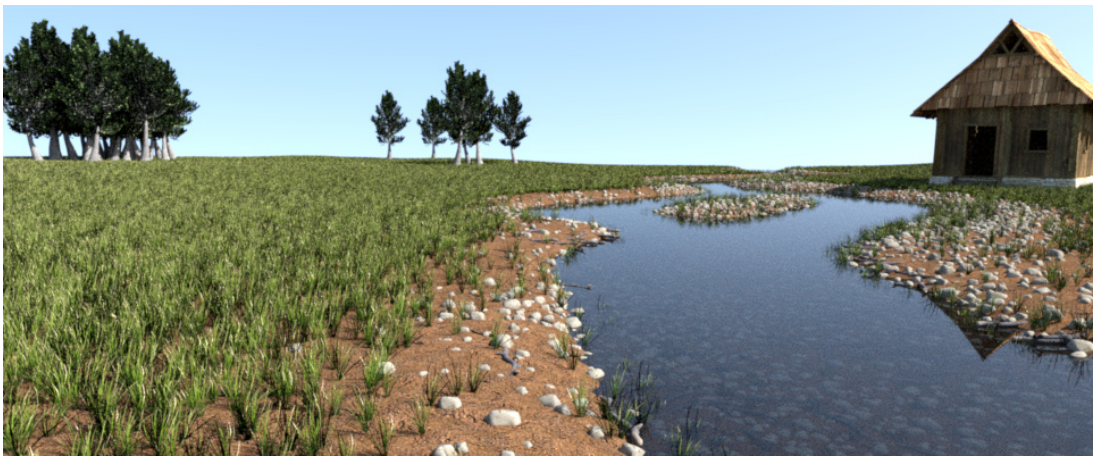


FIGURE 124 : Le système de décorations fantômes semble être parfaitement adapté pour enrichir les scènes virtuelles avec de nombreux détails géométriques.

extensions permettraient la gestion de fonctions de distributions (pour la végétation, les objets naturels, *etc*).

La gestion des textures et des matériaux constitue une autre piste intéressante. Ces textures peuvent correspondre à des déformations de la surfaces (réelles ou simulées à l'éclairage), à la couleur de l'objet ou à des textures permettant d'indiquer le type de matériaux de surface utilisés. Toutes ces informations peuvent être gérées de deux manières sur un maillage : soit par un placage manuel (qui demande une paramétrisation de la surface) soit par un positionnement absolu et souvent procédural. Le modèle d'arbre actuel ne prend pas en compte de notion de textures et les matériaux sont visualisés sous la forme d'un champ de couleur uni.

Enfin la gestion des transitions apparaît également comme un défi à relever. En effet, un terrain réel est composé de nombreux éléments hétérogènes, très différents par leur taille et leur nature. La possibilité d'opérer des transitions en douceur mais sans sacrifier le réalisme visuel, permettrait de détailler le sol des terrains avec des branches, des feuilles et des pierres. Une solution que nous avons commencé à étudier consiste en un système de décoration fantôme par pavage (Fig. 124). Ce système peut être associé à la représentation vectorielle d'un monde sous forme d'arbre de construction (l'arbre décrit le monde, le système de décoration le réalise).

Toutes ces améliorations aurait pour effet de renforcer l'expressivité et le réalisme de ce modèle de représentation de terrains par arbres de construction.

Contrôle du modèle

Permettre un contrôle intuitif et de haut-niveau est devenu un objectif de recherche ce dont témoignent de nombreuses publications récentes en modélisation géométrique. Certains modèles, qui ont parfois des défauts (*e. g.* les voxels et leur mauvaise efficacité mémoire ou les effets procéduraux de nature très chaotique et peu intuitifs à manipuler) deviennent populaires grâce à des logiciels qui proposent de très bons outils d'interaction (*e. g.* ZBRUSH [302] pour les voxels ou HOUDINI [355] pour les effets procéduraux). Une des conditions nécessaire à un bon contrôle

est d'avoir la possibilité de manipuler le modèle en temps-réel. C'est pourquoi, optimiser les calculs d'évaluation du modèle d'arbre (et donc l'extraction du relief) est une priorité.

Le contrôle peut être apporté par différents outils : pour le modèle de terrains par arbre de construction, un éditeur interactif et conçu pour le placement et l'édition de primitives offrirait une base solide aux infographistes pour tester le modèle. Le prototype que nous avons mis en place montre qu'il y a un gain de temps conséquent quand on crée une scène de petite taille. À terme, il serait même envisageable d'utiliser un modèle de primitives très simples pour réaliser une application ludique en réalité virtuelle (Fig. 125).

FIGURE 125 : « Bac-à-sable » augmenté [320].

L'utilisation d'outils de plus hauts niveaux, pour créer des environnements plus vastes, serait un bon moyen de combiner génération procédurale et contrôle utilisateur. La possibilité de peindre des zones de plaines et de montagnes, avec une procédure de placement de primitives automatiques et de raccord (pour les rivières par exemple) réduirait le temps d'édition d'un ordre de grandeur.

Notre algorithme de génération procédurale de terrains aurait tout à gagner d'une amélioration du contrôle. Ainsi, la création du réseau hydrographique pourrait être davantage guidée, et le raffinement des trajectoires, au lieu d'être largement automatique, pourrait être contrôlé plus finement de manière manuelle par l'infographiste.

Modélisation procédurale inverse

Enfin, **le défi majeur à relever concerne la modélisation inverse**. Ce problème est extrêmement compliqué : il n'existe que quelques rares modèles d'analyse d'objets pour retrouver les paramètres de génération (en particulier dans les cas où on utilise des grammaires ou des L-systèmes). Ce verrou limite encore l'utilisation de notre modèle. Mais si nous parvenons à transformer un modèle de terrains échantillonné en un arbre de construction relativement efficace, nous ouvrons la voie à de nombreuses autres perspectives.

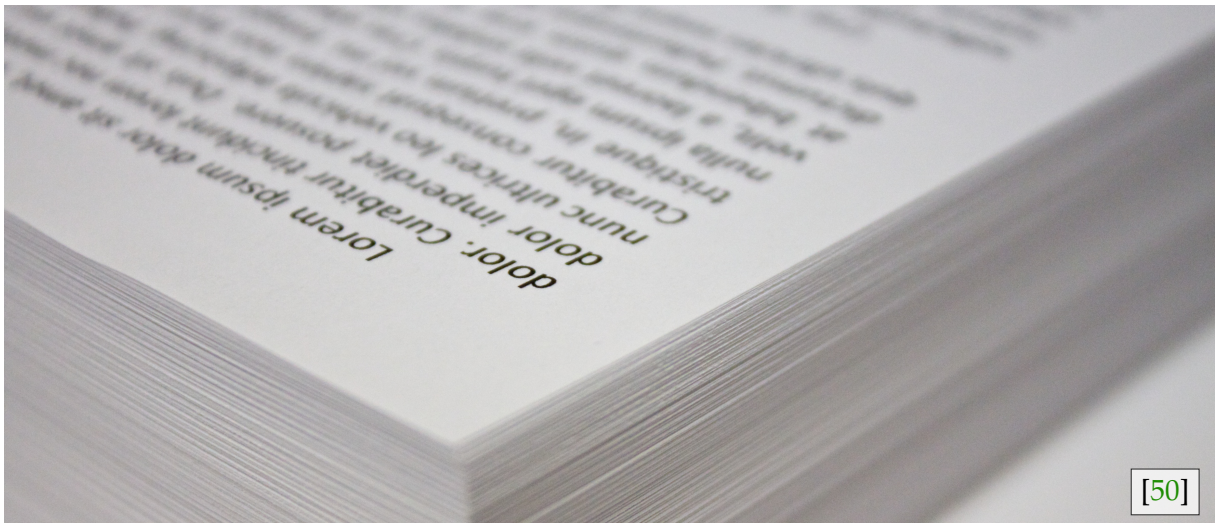
La possibilité de passer d'un type de modèle à un autre autoriserait l'utilisation de procédures complexes ne fonctionnant qu'avec une discrétisation du terrain, comme les simulations d'érosion. L'utilisateur n'aurait qu'à sélectionner une région à échantillonner puis utiliserait des outils d'érosion ou de sculpture manuelle avant de reconvertir sa parcelle de terrain et de la réintégrer dans une scène plus complexe.

L'utilisation de procédures inverses aurait également des impacts sur la compression des données. Un terrain de plusieurs dizaines de giga-octets serait converti en un arbre contenant sa description vectorielle qui ne pèserait que quelques centaines de mega-octets. On peut même espérer utiliser ce modèle pour représenter des planètes entières.

Ainsi, la mise au point d'un processus de modélisation procédurale inverse pourrait avoir un impact important dans plusieurs domaines de la génération procédurale, en particulier pour

les méthodes se basant sur des constructions par morceaux ou sur de la sémantique. Au bout de trois ans de travaux, **c'est cette voie qui nous paraît la plus importante et la plus riche en promesses**.

Bibliographie



Tu as sans doute été une excellente disciple, Oroshi Melicerte : sobre, avide d'apprendre, intelligente. Mais les meilleurs disciples sont ceux qui trahissent. Toi, tu n'as encore trahi rien ni personne. Tu apprends encore des livres, c'est dire.

La Horde du Contrevent — Alain Damasio

[1] Lighthouse 3d, **Terrain generation tutorial : Hill algorithm**. *Site Web* `http://www.stuffwithstuff.com/robot-frog/3d/hills/hill.html`, 2002. (visité le 05/2015). (Cité page 44.)

[2] Allegorithmic, **Substance designer 5**. *Logiciel* `https://www.allegorithmic.com/`, 2015. (Cité page 6.)

[3] Johan Andersson, **5 major challenges in real-time rendering**. *Site Web* `http://www.frostbite.com/wp-content/uploads/2013/05/Bps12_5MajorChallenges_Andersson.pdf`, 2013. (visité le 05/2015). (Cité page 4.)

[4] Bruno Andreotti, Philippe Claudin et Olivier Pouliquen, **Aeolian sand ripples : experimental study of fully developed states**. *Physical review letters*, 96(2):028001 :1–028001 :4, 2006. (Cité page 55.)

[5] Nathan Andrysco, Bedřich Beneš et Matthew Brisbin, **ermeable and absorbent materials in fluid simulations**. *Poster at Symposium on Computer Animation – SCA*, 2008. (Cité page 54.)

[6] Nguyen Hoang Anh, Alexei Sourin et Parimal Aswani, **Physically based hydraulic erosion simulation on graphics processing unit**. *Proc. of the International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia – GRAPHITE*, pages 257–264, Perth, Australia, 2007. ACM. (Cité page 52.)

[7] Anonyme, **Ciel mon doctorat !** *Site Web* `http://cielmondoctorat.tumblr.com/`, 2012. (visité le 08/2015). (Cité page ix.)

[8] ApolloIsMyCoPilot, **Let's play ! mass effect - s26 p4 - even more sidequests... and mako shenanigans !** *Site Web* `https://www.youtube.com/watch?v=WX5K5NjX5YA`, 2013. (visité le 05/2015). (Cité page 27.)

[9] Appeal, **Outcast**. *Jeu-vidéo* `http://www.mobygames.com/game/windows/outcast`, 1999. Infogrames. (Cité page 17.)

[10] Daniel A Ashlock, Stephen P. Gent et Kenneth M. Bryden, **Evolution of l-systems for compact virtual landscape generation**. *Proc. of the Congress of Evolutionary Computation – CEC*, pages 2760–2767, Edinburgh, UK, 2005. IEEE. (Cité pages 36 et 41.)

[11] Daniel A. Ashlock, Stephen P. Gent et Kenneth M. Bryden, **Embryogenesis of artificial landscapes**. *Design by Evolution*, Natural Computing Series, chapitre 12, pages 203–221. Springer, 2008. (Cité pages 36 et 41.)

[12] Samuel Atlan et Michael Garland, **Interactive multiresolution editing and display of large terrains**. *Computer Graphics Forum – CGF*, 25(2):211–223, 2006. (Cité page 16.)

[13] Aymeric Augustin, **Flow-noise en temps réel**. Rapport de stage, École Polytechnique, 2006. (Cité page 39.)

[14] Shai Avidan et Ariel Shamir, **Seam carving for content-aware image resizing**. *Transaction on Graphics – TOG*, 26(3), 2007. (Cité page 64.)

[15] Jeroen Backs, **Realistic (kinda) low poly (sorta) canyon road (definitely) !** *Site Web* `http://www.jeroenbackx.com/index.php/2013/03/canyonroad/`, 2013. (visité le 05/2015). (Cité page 21.)

[16] Shaun Bangay, David de Bruyn et Kevin Glass, **Minimum spanning trees for valley and ridge characterization in digital elevation maps**. *Proc. of the International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa – AFRIGRAPH*, pages 73–82, Franschhoek, South Africa, 2010. ACM. (Cité page 47.)

[17] Aurélien Barbier, Éric Galin et Samir Akkouche, **Complex skeletal implicit surfaces with levels of detail**. *Journal of WSCG – WSCG*, 2004. (Cité page 89.)

[18] James M. Bardeen, **Panorama user's manual**. Rapport technique, Mojoworld, 1992. (Cité page 46.)

[19] Michael F. Barnsley, **The Science of Fractal Images**. Elsevier, 2 édition, 1993. (Cité page 33.)

[20] Michael F. Barnsley, Robert L. Devaney, Benoît B. Mandelbrot, Heiz-Otto Peitgen, Dietmar Saupe, Richard F. Voss, Yuval Fisher et Michael McGuire, **The Science of Fractal Images**. Springer, 1 édition, 1988. (Cité page 33.)

[21] Matthew Beardall, McKay Farley, D. Ouderkirk, C. Reimschussel, J. Smith, M. Jones et P. Egbert, **Goblins by spheroidal weathering**. *Proc. of the Eurographics Workshop on Natural Phenomena – EGWNP*, pages 7–14, Prague, Czech Republic, 2007. Eurographics Association. (Cité pages 57 et 58.)

[22] Farès Belhadj, **Modélisation automatique de géo-environnements naturels et multi-urbains**. Thèse de doctorat, Université Paris 8, 2007. (Cité page 36.)

[23] Farès Belhadj, **Terrain modeling : a constrained fractal model**. *Proc. of the International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa – AFRIGRAPH*, pages 197–204, Grahamstown, South Africa, 2007. ACM. (Cité page 36.)

[24] Farès Belhadj et Pierre Audibert, **Modeling landscapes with ridges and rivers**. *Proc. of the Symposium on Virtual Reality Software and Technology – VRST*, pages 151–154, Monterey, USA, 2005. ACM. (Cité page 45.)

[25] Farès Belhadj et Pierre Audibert, **Modeling landscapes with ridges and rivers : Bottom up approach**. *Proc. of the International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia – GRAPHITE*, pages 447–450, Dunedin, New Zealand, 2005. ACM. (Cité page 45.)

[26] Morten Bendiksen, **Rapid modeling of geology**. Mémoire de Master, University of Bergen, 2013. (Cité page 62.)

[27] Morten Bendiksen, **Rapid modeling of geology**. *Proc. of the Central European Seminar on Computer Graphics – CESCG*, Smolenice, Slovakia, 2013. (Cité page 62.)

[28] Bedřich Beneš, **Real-time erosion using shallow water simulation**. *Proc. of Virtual Reality Interactions and Physical Simulations – VRIPHYS*, Dublin, Ireland, 2007. Eurographics Association. (Cité page 53.)

[29] Bedřich Beneš et Xabier Arriaga, **Table mountains by virtual erosion**. *Proc. of the Eurographics Workshop on Natural Phenomena – EGWNP*, pages 33–39, Dublin, Ireland, 2005. Eurographics Association. (Cité pages 49, 50, et 91.)

[30] Bedřich Beneš et Rafael Forsbach, **Layered data representation for visual simulation of terrain erosion**. *Proc. of the Spring Conference on Computer Graphics – SSCG*, pages 80–85, Washington D.C., USA, 2001. IEEE. (Cité pages 23, 49, 50, et 97.)

[31] Bedřich Beneš et Rafael Forsbach, **Parallel implementation of terrain erosion applied to the surface of mars**. *Proc. of the International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa – AFRIGRAPH*, pages 53–57, Camps Bay, South Africa, 2001. ACM. (Cité page 50.)

[32] Bedřich Beneš et Rafael Forsbach, **Visual simulation of hydraulic erosion**. *Journal of WSCG – WSCG*, 10(1-3):79–86, 2002. (Cité page 52.)

[33] Bedřich Beneš et Tony Roa, **Simulating desert scenery**. *Journal of WSCG – WSCG*, pages 17–22, 2004. (Cité page 55.)

[34] Bedřich Beneš, Václav Těšínský, Jan Hornyš et Sanjiv K. Bhatia, **Hydraulic erosion**. *Computer Animation and Virtual Worlds – CAVW*, 17(2):99–108, 2006. (Cité pages 10, 52, et 53.)

[35] Adrien Bernhardt, André Maximo, Luiz Velho, Houssam Hnaidi et Marie-Paule Cani, **Real-time terrain modeling using CPU-GPU coupled computation**. *Proc. of the Conference on Graphics, Patterns and Images – SIBGRAPI*, pages 64–71, Maceió, Brazil, 2011. IEEE. (Cité page 62.)

[36] Adrien Bernhardt, André Maximo, Luiz Velho, Houssam Hnaidi et Marie-Paule Cani, **Real-time terrain modeling using CPU-GPU coupled computation**. *Poster at SIGGRAPH*, 2011. (Cité page 62.)

[37] Richard Bézin, **Simulation d'évolution topologique : cas de l'interaction fluide / solide**. Thèse de doctorat, Université de Limoges, 2013. (Cité pages 22, 23, et 54.)

[38] Richard Bézin, Benoît Crespin, Xavier Skapin, Olivier Terraz et Philippe Meseure, **Generalized maps for erosion and sedimentation simulation**. *Computers & Graphics – C&G*, 45:1–16, 1996. (Cité page 54.)

[39] Richard Bézin, Benoît Crespin, Xavier Skapin, Olivier Terraz et Philippe Meseure, **Opérations topologiques pour la géomorphologie**. *Proc. of the Journées de l'Association Française d'Informatique Graphique – AFIG*, Biarritz, France, 2011. IRIT Presse. (Cité page 22.)

[40] Richard Bézin, Benoît Crespin, Xavier Skapin, Olivier Terraz et Philippe Meseure, **Topological operations for geomorphological evolution**. *Proc. of Virtual Reality Interactions and Physical Simulations – VRIPHYS*, pages 139–148, Lyon, France, 2011. Eurographics Association. (Cité page 22.)

[41] Richard Bézin, Benoît Crespin, Xavier Skapin, Olivier Terraz et Philippe Meseure, **Programming topological operations for visual illustrations in an introductory geomorphology course**. *Proc. of Eurographics Education Papers – EG Edu*, pages 1–7, Strasbourg, France, 2014. Eurographics Association. (Cité pages 22 et 65.)

[42] Richard Bézin, Alexandre Peyrat, Benoît Crespin, Olivier Terraz, Xavier Skapin et Philippe Meseure, **Interactive hydraulic erosion using cuda**. *Proc. of the International Conference on Computer Vision and Graphics – ICCVG*, pages 225–232, Varsovie, Poland, 2010. Springer. (Cité page 54.)

[43] Richard Bézin, Alexandre Peyrat, Benoît Crespin, Olivier Terraz, Xavier Skapin et Philippe Meseure, **Interactive hydraulic erosion using CUDA**. *Machine Graphics & Vision – MG&V*, 20(2), 2011. (Cité page 54.)

[44] Rafael Bidarra, Klaas Jan De Kraker, Ruben M. Smelik et Tutenel Tim, **Integrating semantics and procedural generation : key enabling factors for declarative modeling of virtual worlds**. *Proc. of FOCUS K3D Conference on Semantic 3D Media and Content*, pages 51–55, Sophia Antipolis, France, 2010. (Cité page 63.)

[45] BigPark, **Joy ride studios**. *Jeu-vidéo* http://en.wikipedia.org/wiki/Joy_Ride_Turbo, 2012. Microsoft. (Cité page 21.)

[46] Jonathan Blow, **Krigin is cool**. *Site Web* http://the-witness.net/news/2010/05/kriging-is-cool/, 2010. (visité le 05/2015). (Cité page 31.)

[47] Florian Boesch, **Webgl gpu landscaping and erosion**. *Site Web* http://codeflow.org/entries/2011/nov/10/webgl-gpu-landscaping-and-erosion/, 2011. (visité le 05/2015). (Cité page 55.)

[48] Geoff Bohling, **Kriging**. Rapport technique, University of Kansas, 2005. (Cité page 31.)

[49] Micah Bojrab, Michel Abdul-Massih et Bedřich Beneš, **Perceptual importance of lighting phenomena in rendering of animated water**. *Transaction on Applied Perception – TAP*, 10(1):2 :1–2 :18, 2013. (Cité page 10.)

[50] Jonathan Joseph Bondhus, **A stack of copy paper**. *Modified File under CC BY-SA 3.0 license* http://commons.wikimedia.org/wiki/File:Stack_of_Copy_Paper.jpg., 2011. (Cité page 171.)

[51] Jean-Jacques Bourdin, **Jeux de roles**. Maîtrise de théâtre, Inconnu, 1997. http://www.ai.univ-paris8.fr/~jj/Pers/JDR/Maitr.2.pdf. (Cité page viii.)

[52] Paul Bourke, **Frequency synthesis of landscapes (and clouds)**. *Site Web* http://paulbourke.net/fractals/noise/, 1997. (visité le 05/2015). (Cité page 38.)

[53] Paul Bourke, **Modelling fake planets**. *Site Web* http://paulbourke.net/fractals/noise/, 2000. (visité le 05/2015). (Cité page 43.)

[54] Paul Bourke, **Dla - diffusion limited aggregation**. *Site Web* http://paulbourke.net/fractals/dla/, 2014. (visité le 05/2015). (Cité page 162.)

[55] Gwyneth Bradbury, Il Choi, Cristina Amati, Kenny Mitchell et Tim Weyrich, **Frequency-based creation and editing of virtual terrain**. *Proc. of European Conference on Visual Media Production – CVMP*, London, UK, 2014. (Cité page 62.)

[56] Sylvain Brandel, Sébastien Schneider, Michel Perrin, Nicolas Guiard, Jean-Français Rainaud, Pascal Lienhard et Yves Bertrand, **Automatic building of structured geological models**. *Journal of Computing and Information Science in Engineering*, 5(2):138–148, 2005. (Cité pages 22, 23, et 65.)

[57] Amanda Briney, **Stream order**. *Site Web* http://geography.about.com/od/physicalgeography/a/streamorder.htm, 2015. (visité le 05/2015). (Cité page 132.)

[58] Cyril Briquet, **Génération et affichage temps réel de terrain sur carte graphique**. Mémoire de Master, Université Lumière Lyon 2, 2014. (Cité page 105.)

[59] John Brosz, Faramarz F. Samavati et Mario Costa Sousa, **Terrain synthesis by-example**. *Advances in Computer Graphics and Computer Vision*, 4:58–77, 2007. (Cité page 59.)

[60] Éric Bruneton et Fabrice Neyret, **Real-time rendering and editing of vector-based terrains**. *Computer Graphics Forum – CGF*, 27(2):311–320, 2008. (Cité pages 63 et 64.)

[61] Tim Burrell, **Advected river textures**. Mémoire de Master, Dalhousie University, 2008. (Cité page 48.)

[62] Angel Fernandez Cabezas et Tommy Thompson, **Real-time procedural terrain generation through swarm behaviours**. *Poster at the International Conference on Foundations of Digital Games – FDG*, 2013. (Cité pages 41 et 42.)

[63] Brian Cabral et Leith Casey Leedom, **Imaging vector fields using line integral convolution**. *Proc. of SIGGRAPH Conference – SIGGRAPH*, pages 263–270, Anaheim, USA, 1993. ACM. (Cité page 39.)

[64] 22 Cans, **Godus**. *Jeu-vidéo* `http://en.wikipedia.org/wiki/Godus`, 2013. 22 Cans. (Cité page 19.)

[65] John Carmack, **Principles of lighting and rendering with john carmack at quakecon**. *Site Web* `https://youtu.be/IyUgHPs86XM?t=4461`, 2013. (visité le 05/2015). (Cité page 26.)

[66] Guillaume Caumon, Pauline Collon-Drouaillet, Christian Le Carlier De Veslud, Sophie Viseur et Judith Sausse, **Surface-based 3d modeling of geological structures**. *Mathematical Geosciences*, 41(8):927–945, 2009. (Cité page 65.)

[67] Miguel Cepero, **Voxel farm engine**. *Logiciel* `http://voxelfarm.com/`, 2014. (Cité page 6.)

[68] Jorge Cham, **Piled higher and depper (phd comics)**. *Site Web* `http://phdcomics.com/comics.php`, 1997. (visité le 08/2015). (Cité page ix.)

[69] Benoit Chanclou, Annie Luciani et Arash Habibi, **Physical models of loose soils dynamically marked by a moving object**. *Proc. of the Computer Animation Conference*, pages 27–35, Geneva, Switzerland, 1996. IEEE. (Cité page 50.)

[70] Stephen Chenney, **Flow tiles**. *Proc. of Symposium on Computer Animation – SCA*, pages 233–242, Grenoble, France, 2004. Eurographics Association. (Cité page 48.)

[71] Min-Yu Chiang, Shih-Chun Tu, Jun-Yan Huang, Wen-Kai Tai, Cheng-Duo Liu et Chin-Chen Chang, **Terrain synthesis : An interactive approach**. *Proc. of International Workshop on Advanced Imaging Technology – IWAIT*, Jeju, South Korea, 2005. (Cité page 59.)

[72] Norishige Chiba, Kazunobu Muraoka et K. Fujita, **An erosion model based on velocity fields for the visual simulation of mountain scenery**. *The Journal of Visualization and Computer Animation – JVCA*, 9(4):185–194, 1998. (Cité page 52.)

[73] Colin CHILDS, **Interpolating surfaces in ArcGIS spatial analyst**. Rapport technique, Esri, 2004. (Cité page 32.)

[74] David CLYDE, **Adding realistic rivers to random terrain**. *Site Web* `http://archive.gamedev.net/archive/reference/programming/features/randomriver/`, 2004. (visité le 05/2015). (Cité page 46.)

[75] Jonathan M. COHEN, John F. HUGHES et Robert C. ZELEZNIK, **Harold : A world made of drawings**. *Proc. of the international symposium on Non-photorealistic animation and rendering – NPAR*, pages 83–90, Anaheim, USA, 2000. ACM. (Cité page 60.)

[76] Daybreak Game COMPANY, **Everquest next**. *Jeu-vidéo* `http://en.wikipedia.org/wiki/EverQuest_Next`, en développement. Daybreak Game Company. (Cité page 23.)

[77] Wikipedia CONTRIBUTORS, **Wikipedia, the free encyclopedia : Polygon mesh**. *Site Web* `http://en.wikipedia.org/wiki/Polygon_mesh`, 2004. (visité le 05/2015). (Cité page 21.)

[78] Wikipedia CONTRIBUTORS, **Wikipedia, the free encyclopedia : Geometric design of roads**. *Site Web* `http://en.wikipedia.org/wiki/Geometric_design_of_roads`, 2009. (visité le 05/2015). (Cité page 80.)

[79] Robert L. COOK et Tony DEROSE, **Wavelet noise**. *Transaction on Graphics – TOG*, 24(3):803–811, 2005. (Cité page 39.)

[80] Greg COOMBE, **An introduction to scattered data approximation**. Rapport de recherche, University of North Carolina at Chapel Hill, 2006. (Cité page 28.)

[81] Massimiliano CORSINI, Paolo CIGNONI et Roberto SCOPIGNO, **Efficient and flexible sampling with blue noise properties of triangular meshes**. *Transactions on Visualization and Computer Graphics – TVCG*, 18(6):914–924, 2012. (Cité page 108.)

[82] Bruno Barcellos COUTINHO, Antonio A. F. OLIVEIRA, Yalmar Ponce ATENCIO et Gilson Antonio GIRALDI, **Rain scene animation through particle systems and surface flow simulation by SPH**. *Proc. of the Conference on Graphics, Patterns and Images – SIBGRAPI*, Gramado, Brazil, 2010. IEEE. (Cité pages 47 et 48.)

[83] Benoît CRESPIN, Carole BLANC et Christophe SCHLICK, **Implicit sweep objects**. *Computer Graphics Forum – CGF*, 15(3):165–174, 1996. (Cité page 82.)

[84] Leandro CRUZ, Francisco GANACIM, Djalma LUCIO, Luiz VELHO et Luiz Henrique de FIGUEIREDO, **Exemplar-based terrain synthesis**. *Work In Progress at the Conference on Graphics, Patterns and Images – SIBGRAPI*, Arequipa, Peru, 2013. IEEE. (Cité page 59.)

[85] Leandro CRUZ, Luiz VELHO, Éric GALIN, Peytavie ADRIEN et Éric GUÉRIN, **Patch-based terrain synthesis**. *Poster at the International Conference on Graphics Theory and Applications – GRAPP*, 2015. (Cité page 59.)

[86] Leandro Cruz, Luiz Velho, Djalma Lucio, Éric Galin, Peytavie Adrien et Éric Guérin, **Landscape specification resizing**. *Proc. of the Conferencia Lationoamericana en Informática – CLEI*, Montevideo, Uruguay, 2014. (Cité page 64.)

[87] Crytek, **Cryengine 3 – voxel tool**. *Site Web* https://www.youtube.com/watch?v=-HsuXB6FT0o, 2012. (visité le 05/2015). (Cité page 101.)

[88] Carsten Dachsbacher, **Interactive Terrain Rendering : Towards Realism with Procedural Models and Graphics Hardware**. Thèse de doctorat, Universität Erlangen–Nürnberg, 2006. (Cité page 59.)

[89] Carsten Dachsbacher, Martin Meyer et Marc Stamminger, **Heightfield synthesis by non-parametric sampling**. *Proc. of Vision Modeling and Visualization – VMV*, pages 297–302, Erlangen, Germany, 2005. (Cité page 59.)

[90] Carsten Dachsbacher et Marc Stamminger, **Rendering procedural terrain by geometry image warping**. *Proc. of the Eurographics Symposium on Rendering – EGSR*, pages 103–110, Norrköping, Sweden, 2004. Eurographics Association. (Cité pages 64 et 65.)

[91] Donato D'Ambrosio, S. Di Gregorio, Salvatore Gabriele et Roberto Gaudio, **A cellular automata model for soil erosion by water**. *Journal of Physics and Chemistry of the Earth*, 26(1):33–39, 2001. (Cité page 52.)

[92] Guillaume Damiand et Pascal Lienhardt, **Combinatorial Maps : Efficient Data Structures for Computer Graphics and Image Processing**. CRC Press, 2014. (Cité page 22.)

[93] Daniel Michelon De Carli, Fernando Bevilacqua, Cesar Tadeu Pozzer et Marcos Cordeiro d'Ornellas, **A survey of procedural content generation techniques suitable to game development**. *Proc. of the Brazilian Symposium on Games and Digital Entertainment – SBGAMES*, pages 26–35, Salvador, Brazil, 2011. IEEE. (Cité page 11.)

[94] Daniel Michelon De Carli, Cesar Tadeu Pozzer, Victor Schetinger et Fernando Bevilacqua, **Procedural generation of 3d canyons**. *Proc. of the Conference on Graphics, Patterns and Images – SIBGRAPI*, pages 103–110, Rio de Janeiro, Brazil, 2014. IEEE. (Cité page 46.)

[95] Giliam J. P. de Carpentier, **Interactively synthesizing and editing virtual outdoor terrain**. Mémoire de Master, Delft University of Technology, 2007. (Cité pages 40 et 62.)

[96] Giliam J. P. de Carpentier et Rafael Bidarra, **Interactive gpu-based procedural heightfield brushes**. *Proc. of the International Conference on Foundations of Digital Games – FDG*, pages 55–62, Orlando, USA, 2009. ACM. (Cité pages 10, 40, 61, 62, 75, et 100.)

[97] Wim de Leeuw et Robert van Liere, **Comparing lic and spot noise**. *Proc. of the Conference on Visualization*, pages 359–365, Durham, USA, 1998. IEEE. (Cité page 39.)

[98] Evgenij Derzapf, Björn Ganster, Michael Guthe et Reinhard Klein, **River networks for instant procedural planets**. *Computer Graphics Forum – CGF*, 30(7):2031–2040, 2011. (Cité pages 36 et 46.)

[99] A. R. Dixon, G. H. Kirby et Derek P. M. Wills, **A data structure for artificial terrain generation**. *Computer Graphics Forum – CGF*, 13(1):37–48, 1994. (Cité pages 18 et 36.)

[100] Jonathon Doran et Ian Parberry, **Controlled procedural terrain generation using software agents**. *Transactions on Computational Intelligence and AI in Games – T-CIAIG*, 2(2):111–119, 2010. (Cité pages 41 et 42.)

[101] Julie Dorsey, Alan Edelman, Henrik Wann Jensen, Justin Legakis et Hans Køhling Pedersen, **Modeling and rendering of weathered stone**. *Proc. of SIGGRAPH Conference – SIGGRAPH*, pages 225–234, Los Angeles, USA, 1999. ACM. (Cité pages 57 et 58.)

[102] Vladimir Alves dos Passos et Takeo Igarashi, **Landsketch : A first person point-of-view example-based terrain modeling approach**. *Proc. of the International Symposium on Sketch-Based Interfaces and Modeling – SBIM*, pages 61–68, Anaheim, USA, 2013. ACL. (Cité pages 60 et 61.)

[103] Jonathan Dummer, **Cone step mapping : An iterative ray-heightfield intersection algorithm**. *Site Web* `http://lonesock.net/files/ConeStepMapping.pdf`, 2006. (visité le 05/2015). (Cité page 113.)

[104] Thomas Dunne, **Water in environmental planning**. Macmillan, 1978. (Cité page 147.)

[105] Jonathan Dupuy, Jean-Claude Iehl et Pierre Poulin, **Quadtrees on the GPU**. *GPU Pro*^{5} *: Advanced Rendering Techniques*, volume 5 de *GPU Pro*, pages 439–450. CRC Press, 2014. (Cité pages 106 et 124.)

[106] David S. Ebert, Forest Kenton Musgrave, Darwyn Peachey, Ken Perlin et Steven Worley, **Texturing and Modeling : A Procedural Approach**. The Morgan Kaufmann Series in Computer Graphics. Elsevier, 3rd édition, 1998. (Cité pages 38, 40, et 112.)

[107] Alexei A. Efros et William T. Freeman, **Image quilting for texture synthesis and transfer**. *Proc. of SIGGRAPH Conference – SIGGRAPH*, pages 341–346, Los Angeles, USA, 2001. ACM. (Cité page 59.)

[108] Jean-François ElHajjar, **Simulation d'écoulements liquides sur des surfaces solides pour l'animation en synthèse d'images**. Thèse de doctorat, Université de Limoges, 2008. (Cité page 47.)

[109] Hugo Elias, **Spherical landscapes**. *Site Web* `http://freespace.virgin.net/hugo.elias/models/m_landsp.htm`, 2000. (visité le 05/2015). (Cité page 43.)

[110] Arnaud Emilien, **Création interactive de mondes virtuels**. Thèse de doctorat, Université de Montréal / Université de Grenoble, 2014. (Cité page 46.)

[111] Arnaud Emilien, Pierre Poulin, Marie-Paule Cani et Ulysse Vimont, **Design de cascades réalistes : une méthode pour combiner contrôle interactif et modèle procédural**. *Revue Electronique Francophone d'Informatique Graphique – REFIG*, 8(1):33–44, 2014. (Cité pages 46 et 63.)

B

[112] Arnaud Emilien, Pierre Poulin, Marie-Paule Cani et Ulysse Vimont, **Interactive procedural modelling of coherent waterfall scenes**. *Computer Graphics Forum – CGF*, 2014. (Cité pages 45, 46, et 63.)

[113] Arnaud Emilien, Ulysse Vimont, Marie-Paule Cani, Pierre Poulin et Bedřich Beneš, **Worldbrush : Interactive example-based synthesis of procedural virtual worlds**. *Transaction on Graphics – TOG*, 2015. (Cité page 64.)

[114] T. Endreny, **Fluvial geomorphology module : Classification systems**. *Site Web* `http://www.fgmorph.com/fg_4_1.php`, 2010. (visité le 05/2015). (Cité page 130.)

[115] Andy Farnell, **Factors against procedural audio**. *Site Web* `http://www.obiwannabe.co.uk/html/papers/proc-audio/node21.html`, 2007. (visité le 05/2015). (Cité page 27.)

[116] Paul Fearing, **Computer modelling of fallen snow**. *Proc. of SIGGRAPH Conference – SIGGRAPH*, pages 37–46, New Orleans, USA, 2000. ACM. (Cité page 56.)

[117] António Ramires Fernandes, **Particle deposition**. *Site Web* `http://www.lighthouse3d.com/opengl/terrain/index.php3?particle`, 2001. (visité le 05/2015). (Cité page 44.)

[118] Niels v. Festenberg et Stefan Gumhold, **A geometric algorithm for snow distribution in virtual scenes**. *Proc. of the Eurographics Workshop on Natural Phenomena – EGWNP*, pages 15–22, Munich, Germany, 2009. Eurographics Association. (Cité page 56.)

[119] Niels v. Festenberg et Stefan Gumhold, **Diffusion-based snow cover generation**. *Computer Graphics Forum – CGF*, 30(6):1837–1849, 2011. (Cité page 56.)

[120] David Foldes et Bedřich Beneš, **Occlusion-based snow accumulation simulation**. *Proc. of Virtual Reality Interactions and Physical Simulations – VRIPHYS*, pages 35–41, Dublin, Ireland, 2007. Eurographics Association. (Cité pages 56 et 57.)

[121] Alain Fournier, Don Fussell et Loren Carpenter, **Comment on computer rendering of stochastic models**. *Communications of the ACM*, 25(8):581–584, 1982. (Cité page 35.)

[122] Alain Fournier, Don Fussell et Loren Carpenter, **Computer rendering of stochastic models**. *Computer Graphics*, 25(6):371–384, 1982. (Cité page 35.)

[123] Miguel Monteiro de Sousa Frade, **Evolving Artificial Terrains with Automated Genetic Terrain Programing**. Thèse de doctorat, Universidad de Extremadura, 2012. (Cité page 41.)

[124] Miguel Monteiro de Sousa Frade, Francisco Fernàndez de Vega et Carlos Cotta, **Genetic terrain programming : an aesthetic approach to terrain generation**. *Proc. of Computer Games Multimedia and Allied Technology – CGAT*, Singapore, Singapore, 2008. GSTF Digital Library. (Cité pages 41 et 42.)

[125] Miguel Monteiro de Sousa Frade, Francisco Fernàndez de Vega et Carlos Cotta, **Modelling video games' landscapes by means of genetic terrain programming-a new approach for improving users' experience**. *Applications of evolutionary computing*, pages 485–490, 2008. (Cité page 41.)

[126] Miguel Monteiro de Sousa Frade, Francisco Fernàndez de Vega et Carlos Cotta, **Adding zoom feature to terrain programmes**. *Proc. of the Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados*, Málaga, Spain, 2009. (Cité page 41.)

[127] Miguel Monteiro de Sousa Frade, Francisco Fernàndez de Vega et Carlos Cotta, **Breeding terrains with genetic terrain programming : The evolution of terrain generators**. *International Journal of Computer Games Technology – IJCGT*, 2009. (Cité page 41.)

[128] Miguel Monteiro de Sousa Frade, Francisco Fernàndez de Vega et Carlos Cotta, **Evolution of artificial terrains for video games based on accessibility**. *Applications of evolutionary computation*, 2010. (Cité page 41.)

[129] Miguel Monteiro de Sousa Frade, Francisco Fernàndez de Vega et Carlos Cotta, **Aesthetic terrain programs database for creativity assessment**. *Proc. of the Conference on Computational Intelligence and Games – IG*, Granada, Spain, 2012. IEEE. (Cité page 41.)

[130] Miguel Monteiro de Sousa Frade, Francisco Fernàndez de Vega et Carlos Cotta, **Automatic evolution of programs for procedural generation of terrains for video games**. *Soft Computing*, 16(11):1893–1912, 2012. (Cité page 41.)

[131] Jérémy Gaillard, **Un modèle hydrologique hiérarchique pour la synthèse de terrains**. Mémoire de Master, INSA de Lyon, 2014. (Cité pages 10, 47, 138, 158, et 161.)

[132] James Gain, Bruce Merry et Patrick Marais, **Parallel, realistic and controllable terrain synthesis**. *Computer Graphics Forum – CGF*, 34(2), 2015. (Cité pages 59, 60, et 61.)

[133] James E. Gain, Patrick Marais et Wolfgang Strasser, **Terrain sketching**. *Proc. of the Symposium on Interactive 3D Graphics and Games – I3D*, pages 31–38, Boston, USA, 2009. ACM. (Cité page 61.)

[134] Andy Gainey, **Procedural planet generation**. *Site Web* `https://experilous.com/1/blog/post/procedural-planet-generation`, 2014. (visité le 05/2015). (Cité pages 44 et 45.)

[135] Bruno Galerne, Ares Lagae, Sylvain Lefebvre et George Drettakis, **Gabor noise by example**. *Transaction on Graphics – TOG*, 31(4):73 :1–73 :9, 2012. (Cité page 59.)

[136] Gas Powered Games, **Supreme commander**. *Jeu-vidéo* `http://www.mobygames.com/game/supreme-commander`, 2007. THQ. (Cité page 17.)

[137] Manuel N. Gamito et Steve C. Maddock, **Ray casting implicit fractal surfaces with reduced affine arithmetic**. *The Visual Computer – TVC*, 23(3):155–165, 2007. (Cité page 112.)

[138] Manuel N. Gamito et Kenton Forest Musgrave, **Procedural landscapes with overhangs**. *Proc. of the Portuguese Computer Graphics Meeting*, Lisbon, Portugal, 2001. (Cité page 16.)

[139] Jean-David Génevaux, Éric Galin, Éric Guérin, Adrien Peytavie et Bedřich Beneš, **Génération procédurale de rivières et de terrains**. *Proc. of the Journées de l'Association Française d'Informatique Graphique – AFIG*, Calais, France, 2012. IRIT Presse. (Cité pages v, 47, 97, 129, et 164.)

[140] Jean-David Génevaux, Éric Galin, Éric Guérin, Adrien Peytavie et Bedřich Beneš, **Terrain generation using procedural models based on hydrology**. *Transaction on Graphics – TOG*, 32(4):143 :1–143 :13, 2013. (Cité pages v, 47, 97, 129, et 164.)

[141] Jean-David Génevaux, Éric Galin, Adrien Peytavie, Éric Guérin, Cyril Briquet, François Grosbellet et Bedřich Beneš, **Terrain modeling from feature primitives**. *Computer Graphics Forum – CGF*, 2015. (Cité pages v, 69, 105, et 164.)

[142] Jean-David Génevaux, François Grosbellet, Éric Galin, Adrien Peytavie, Éric Guérin, Briquet Cyril et Bedřich Beneš, **Modélisation de terrains par primitives**. *Proc. of the Journées de l'Association Française d'Informatique Graphique – AFIG*, Reims, France, 2014. IRIT Presse. (Cité pages v, 69, 105, et 164.)

[143] Christian Gentil, **Les fractales en synthese d'images : le modele IFS**. Thèse de doctorat, Université Claude Bernard Lyon 1, 1992. (Cité page 36.)

[144] Geographic Information Technology Training Alliance, **Drainage networks**. *Site Web* `http://www.gitta.info/TerrainAnalyi/en/html/HydroloAppls_learningObject2.html`, 2010. (visité le 05/2015). (Cité page 131.)

[145] Guillaume Gilet, Jean-Michel Dischler et Djamchi Ghazanfarpour, **Multiple kernels noise for improved procedural texturing**. *The Visual Computer – TVC*, 28(6-8):679–689, 2012. (Cité page 59.)

[146] Guillaume Gilet, Basile Sauvage, Kenneth Vanhoey, Jean-Michel Dischler et Djamchid Ghazanfarpour, **Local random-phase noise for procedural texturing**. *Transaction on Graphics – TOG*, 33(6):195 :1–195 :11, 2014. (Cité pages 38, 39, et 59.)

[147] Christopher Gold et Maciej Dakowicz, **Terrain modelling based on contours and slopes**. *Proc. of International Conference on Computational Science – ICCS*, Amsterdam, Netherlands, 2002. (Cité page 19.)

[148] Christopher M. Gold, T. D. Charters et J. Ramsden, **Automated contour mapping using triangular element data structures and an interpolant over each irregular triangular domain**. *Computer Graphics*, 11(2):170–175, 1977. (Cité page 20.)

[149] David Goutx et Stephane Ladreyt, **Hydraulique des cours d'eau : La théorie et sa mise en pratique**. Cours (introduction pédagogique), Centre d'Etudes Techniques Maritimes Et Fluviales, 2001. (Cité page 130.)

[150] David Grelsson, **Tile based procedural terrain generation in real-time**. Mémoire de Master, Blekinge Institute of Technology, 2014. (Cité page 44.)

[151] Eric Guilbert, Julien Gaffuri et Bernhard Jenny, **Terrain generalisation**. *Abstracting Geographic Information in a Data Rich World*, chapitre 8, pages 227–258. Springer, 2014. (Cité page 16.)

[152] Éric Guérin, **Approximation fractale de courbes et de surfaces**. Thèse de doctorat, Université Claude Bernard Lyon 1, 2002. (Cité page 36.)

[153] Stefan Gustavson, **Simplex noise demystified**. Rapport technique, Linköping University, 2005. (Cité page 39.)

[154] Luke Halliwell, **Procedural content generation**. *Site Web* `https://lukehalliwell.wordpress.com/2008/08/05/procedural-content-generation/`, 2008. (visité le 05/2015). (Cité page 25.)

[155] John C. Hart, **Sphere tracing : A geometric method for the antialiased ray tracing of implicit surfaces**. *The Visual Computer – TVC*, 12(10):527–545, 1996. (Cité page 111.)

[156] John C. Hart, **Perlin noise pixel shaders**. *Proc. of the workshop on Graphics Hardware*, pages 31–38, Los Angeles, USA, 2001. ACM/Eurographics. (Cité page 40.)

[157] David J. Heeger et James R. Bergen, **Pyramid-based texture analysis/synthesis**. *Proc. of SIGGRAPH Conference – SIGGRAPH*, pages 229–238, Los Angeles, USA, 1995. ACM. (Cité page 59.)

[158] Quintijn Hendrickx, Ruben Smelik et Rafael Bidarra, **Real-time rendering of river networks**. *Poster at the Symposium on Interactive 3D Graphics and Games – I3D*, 2010. (Cité pages 47 et 48.)

[159] Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden et Alexandru Iosup, **Procedural content generation for games : A survey**. *Transactions on Multimedia Computing, Communications, and Applications – TOMCCAP*, 9(1):1 :1–1 :22, 2013. (Cité page 11.)

[160] Christian Henning et Peter Stephenson, **Accelerating the ray tracing of height fields**. *Proc. of the International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia – GRAPHITE*, pages 254–258, Singapore, Singapore, 2004. ACM. (Cité page 112.)

[161] Tommy Hinks et Ken Museth, **Wind-driven snow buildup using a level set approach**. *Proc. of Eurographics Ireland Workshop Series*, pages 19–26, Dublin, Ireland, 2009. ACM. (Cité page 56.)

[162] Houssam Hnaidi, **Contrôle dans la génération de formes naturelles**. Thèse de doctorat, Université Claude Bernard Lyon 1, 2010. (Cité pages 28, 36, et 62.)

[163] Houssam Hnaidi, Éric Guérin, Samir Akkouche, Adrien Peytavie et Éric Galin, **Feature based terrain generation using diffusion equation**. *Computer Graphics Forum – CGF*, 29(7): 2179–2186, 2010. (Cité pages 28, 61, et 62.)

[164] Kai Hormann, Salvatore Spinello et Peter Schröder, **C^1-Continuous Terrain Reconstruction from Sparse Contours**. *Proc. of Vision Modeling and Visualization – VMV*, pages 289–297, Munich, Germany, 2003. (Cité page 19.)

B

[165] Robert E. Horton, **Erosional development of streams and their drainage basins ; hydrophysical approach to quantitative morphology**. *Geological society of America bulletin*, 56(3):275–370, 1945. (Cité page 140.)

[166] Matej Hudák et Roman Ďurikovič, **Terrain models for mass movement erosion**. *Proc. of Theory and Practice of Computer Graphics – TP.CG*, pages 1–8, Warwick, UK, 2011. Eurographics Association. (Cité page 49.)

[167] Remco Huijser, Jeroen Dobbe, Willem F. Bronsvoort et Rafael Bidarra, **Procedural natural systems for game level design**. *Proc. of the Brazilian Symposium on Games and Digital Entertainment – SBGAMES*, pages 189–198, Florianópolis, Brazil, 2010. (Cité pages 45 et 46.)

[168] Carl Hultquist, James E. Gain et David E. Cairns, **Affective scene generation**. *Proc. of the International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa – AFRIGRAPH*, pages 59–63, Cape Town, South Africa, 2006. (Cité page 63.)

[169] Paul Humphries, **River ecology and research**. *Site Web* `https://paulhumphriesriverecology.wordpress.com/`, 2015. (visité le 05/2015). (Cité page 130.)

[170] Michael F. Hutchinson et John C. Gallant, **Representation of terrain**. *Geographical information systems : Principles and Technical Issues*, volume 1, chapitre 9, pages 105–124. John Wiley and Sons Ltd., 2 édition, 1999. (Cité page 16.)

[171] IBM, **Paperwork explosion**. *Site Web* `http://www.wired.com/2014/03/tech-time-warp-henson-ibm/`, 2014. (visité le 05/2015). (Cité page 2.)

[172] Mathieu Isorce, **Modélisation de terrains hétérogènes avec surplombs**. Mémoire de Master, Université Claude Bernard Lyon 1, 2007. (Cité page 24.)

[173] Tomoya Ito, Tadahiro Fujimoto, Kazunobu Muraoka et Norishige Chiba, **Modeling rocky scenery taking into account joints**. *Proc. of Computer Graphics International – CGI*, pages 244–247, Tokyo, Japan, 2003. IEEE. (Cité page 49.)

[174] Hyatt Moore IV, **Procedural noise - categories**. *Site Web* `http://web.stanford.edu/~hyatt4/research/school_projects/CS448X/Categories.html`, 2011. (visité le 05/2015). (Cité page 39.)

[175] Alex Jarocha-Ernst, **Creating landscapes with simulated colliding plates**. Mémoire de Master, Rochester Institute of Technology, 2006. (Cité page 48.)

[176] Bernhard Jenny, Helen Jenny et Lorenz Hurni, **Terrain generalization with multi-scale pyramids constrained by curvature**. *Cartography and Geographic Information Science*, 38(2): 110–116, 2011. (Cité page 65.)

[177] Helen Jenny, Bernhard Jenny, William E. Cartwright et Lorenz Hurni, **Interactive local terrain deformation inspired by hand-painted panoramas**. *The Cartographic Journal*, 48(1):11–20, 2011. (Cité page 62.)

[178] Balázs Jákó et Balázs Tóth, **Fast hydraulic and thermal erosion on the gpu**. *Proc. of the Central European Seminar on Computer Graphics – CESCG*, Viničné, Slovakia, 2011. (Cité pages 50 et 53.)

[179] Michael D. Jones, McKay Farley, Joseph Butler et Matthew Beardall, **Directable weathering of concave rock using curvature estimation**. *Transactions on Visualization and Computer Graphics – TVCG*, 16(1):81–94, 2010. (Cité page 58.)

[180] Devenda Kalra et Alan Barr, **Guaranteed ray intersections with implicit surfaces**. *Computer Graphics*, 23(3):297–306, 1989. (Cité page 111.)

[181] K. Raiyan Kamal et Yusuf Sarwar Uddin, **Parametrically controlled terrain generation**. *Proc. of the International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia – GRAPHITE*, pages 17–23, Perth, Australia, 2007. ACM. (Cité page 43.)

[182] Béatrice Kammerer, **Je remercie, tu remercies, nous remercions**. *Site Web* `http://jevoudraisremercier.tumblr.com/`, 2014. (visité le 08/2015). (Cité pages ix et x.)

[183] Viktor Kämpe, Erik Sintorn et Ulf Assarsson, **High resolution sparse voxel DAGs**. *Transaction on Graphics – TOG*, 32(4):101 :1–101 :13, 2013. (Cité page 23.)

[184] Benjamin Keinert, Henry Schäfer, Johann Korndörfer, Urs Ganse et Marc Stamminger, **Enhanced sphere tracing**. *Proc. of Smart Tools & Apps for Graphics – STAG*, Cagliari, Italy, 2014. Eurographics Association. (Cité page 113.)

[185] Alex D. Kelley, Michael C. Malin et Gregory M. Nielson, **Terrain simulation using a model of stream erosion**. *Computer Graphics*, 22(4):263–268, 1988. (Cité pages 36 et 45.)

[186] Burno Kemen et Laco Hrabcak, **Outerra**. *Logiciel* `http://www.outerra.com/`, 2010. (Cité page 5.)

[187] Andrew Kensler, Aaron Knoll et Peter Shirley, **Better gradient noise**. Rapport technique, University of Utah, 2008. (Cité page 39.)

[188] Peter Kipfer et Rüdiger Westermann, **Realistic and interactive simulation of rivers**. *Proc. of Graphics Interface – GI*, pages 41–48, Quebec City, Canada, 2006. Canadian Information Processing Society. (Cité page 47.)

[189] Jasper Knight et Stefan W. Grab, **Lightning as a geomorphic agent on mountain summits : evidence from southern africa**. *Geomorphology*, 204:61–70, 2014. (Cité page 58.)

[190] Çetin Koca et Uğur Güdükbay, **A hybrid representation for modeling, interactive editing, and realtime visualization of terrains with volumetric features**. *International Journal of Geographical Information Science*, 28(9):1821–1847, 2014. (Cité page 21.)

[191] Demetris Koutsoyiannis, **The hurst phenomenon and fractional gaussian noise made easy**. *Hydrological Sciences Journal*, 47(4):573–595, 2002. (Cité page 33.)

[192] Peter Krištof, Bedřich Beneš, Jaroslav Křivánek et Ondřej Šťava, **Hydraulic erosion using smoothed particle hydrodynamics**. *Computer Graphics Forum – CGF*, 28(2):219–228, 2009. (Cité page 54.)

[193] Robert Kropf, **Digitales geländemodell**. *Modified File under CC BY-SA 3.0 DE license* http://commons.wikimedia.org/wiki/File:Digitales_Geländemodell.png., 1999. (Cité page 20.)

[194] Michał Kurowski, **Procedural generation of meandering rivers inspired by erosion**. *Journal of WSCG – WSCG*, pages 79–86, 2012. (Cité pages 10, 46, et 54.)

[195] Michał Kurowski, **Procedural generation of meandering rivers with oxbow lakes**. *Short Paper at Computer Graphics International – CGI*, Hannover, Germany, 2013. (Cité pages 46, 54, et 151.)

[196] Michał Kurowski, **Modeling of an evolving river network with meanders, oxbow lakes and river deltas**. Thèse de doctorat, Warsaw University of Technology, 2015. (Cité pages 46, 54, et 151.)

[197] Ares Lagae et Philip Dutré, **A procedural object distribution function**. *Transaction on Graphics – TOG*, 24(4):1442–1461, 2005. (Cité page 152.)

[198] Ares Lagae, Sylvain Lefebvre, George Drettakis et Philip Dutré, **Procedural noise using sparse Gabor convolution**. *Transaction on Graphics – TOG*, 28(3):54 :1–54 :10, 2009. (Cité page 39.)

[199] Ares Lagae, Lefebvre Syvalin, Robert L. Cook, Tony DeRose, Georges Drettakis, David S. Ebert, J. P. Lewis, Ken Perlin et Zwicker M., **A survey of procedural noise functions**. *Computer Graphics Forum – CGF*, 29(8):2579–2600, 2010. (Cité page 39.)

[200] Dominique Laigle, Nohamed Naaim, Pierre Saramito, Fabrice Neyret et Marie-Paule Cani, **Coulées de boue et avalanches virtuelles : un outil visuel de communication et de caractérisation pour les risques naturels en montagne**. *Ingénieries - eau, agriculture, territoires*, pages 127–136, 2003. (Cité page 56.)

[201] Samuli Laine et Tero Karras, **Efficient sparse voxel octrees**. *Proc. of the Symposium on Interactive 3D Graphics and Games – I3D*, pages 55–63, Boston, USA, 2009. ACM. (Cité page 23.)

[202] Samuli Laine et Tero Karras, **Efficient sparse voxel octrees – analysis, extensions, and implementation**. Rapport technique, NVIDIA Corporation, 2010. (Cité page 23.)

[203] Samuli Laine et Tero Karras, **Efficient sparse voxel octrees**. *Transactions on Visualization and Computer Graphics – TVCG*, 17(1):1048–1059, 2011. (Cité page 23.)

[204] Nicholas Lancaster, **Geomorphology of desert dunes**. Routledge, 1995. (Cité page 55.)

[205] Michael S. LANGER et Linquiao ZHANG, **Rendering falling snow using an inverse fourier transform**. *Short at SIGGRAPH Conference – SIGGRAPH*, San Diego, USA, 2003. (Cité page 38.)

[206] Michael S. LANGER, Linquiao ZHANG, Allison W. KLEIN, Aditya BHATIA, Javeen PEREIRA et Dipinder REKHI, **A spectral-particle hybrid method for rendering falling snow**. *Proc. of the Eurographics Symposium on Rendering – EGSR*, volume 4, pages 217–226, Norrköping, Sweden, 2004. Eurographics Association. (Cité page 38.)

[207] LAPIN, **Vie de thesarde / vie de jeune docteure**. *Site Web* `http://lapinobservateur.over-blog.com/`, 2010. (visité le 08/2015). (Cité page ix.)

[208] Steven M. LAVALLE, **Rapidly-exploring random trees : A new tool for path planning**. Rapport technique, Iowa State University, 1998. (Cité page 162.)

[209] Hugo LEDOUX et Christopher GOLD, **An efficient natural neighbour interpolation algorithm for geoscientific modelling**. *Developments in Spatial Data Handling*, pages 97–108. Springer, 2005. (Cité page 32.)

[210] Sylvain LEFEBVRE, **Synthèse de textures par l'exemple pour les applications interactives**. Habilitation à Diriger des Recherches, Université de Lorraine, 2014. (Cité page 59.)

[211] Sylvain LEFEBVRE et Hugues HOPPE, **Parallel controllable texture synthesis**. *Transaction on Graphics – TOG*, 24(3):777–786, 2005. (Cité page 59.)

[212] Pierre-François LÉON, Xavier SKAPIN et Philippe MESEURE, **A topology-based animation model for the description of 2d models with a dynamic structure**. *Proc. of Virtual Reality Interactions and Physical Simulations – VRIPHYS*, pages 67–76, Grenoble, France, 2008. Eurographics Association. (Cité pages 22 et 65.)

[213] John-Peter LEWIS, **Generalized stochastic subdivision**. *Transaction on Graphics – TOG*, 6(3):167–190, 1987. (Cité page 36.)

[214] John-Peter LEWIS, **Algorithms for solid noise synthesis**. *Computer Graphics*, 23(3):263–270, 1989. (Cité page 39.)

[215] John-Peter LEWIS, **Is the fractal model appropriate for terrain ?** Rapport technique, Disney Secret Lab., 1990. (Cité page 34.)

[216] David LI, **Ocean wave simulation**. *Site Web* `http://david.li/waves/`, 2014. (visité le 05/2015). (Cité page 38.)

[217] Xin LI et J. Michael MOSHELL, **Modeling soil : realtime dynamic models for soil slippage and manipulation**. *Proc. of SIGGRAPH Conference – SIGGRAPH*, pages 361–368, Anaheim, USA, 1993. ACM. (Cité page 50.)

[218] Zhilin LI, Zhu QING et Christopher GOLD, **Digital Terrain Modeling : Principles and Methodology**. CRC Press, 2005. (Cité page 16.)

[219] Noise Lib, **Generating coherent noise**. *Site Web* http://libnoise.sourceforge.net/noisegen/, 2007. (visité le 05/2015). (Cité page 39.)

[220] Endre M. Lidal, Morten Bendiksen, Daniel Patel et Ivan Viola, **Rapid sketch-based 3d modeling of geology**. *Proc. of the Workshop on Visualisation in Environmental Sciences – EnvirVis*, pages 1–5, Leipzig, Germany, 2013. Springer. (Cité page 65.)

[221] Endre M. Lidal, H. Hauser et Ivan Viola, **Geological storytelling : Graphically exploring and communicating geological sketches**. *Proc. of the International Symposium on Sketch-Based Interfaces and Modeling – SBIM*, pages 11–20, Annecy, France, 2012. Eurographics Association. (Cité page 65.)

[222] Ives Macêdo, João Paulo Gois et Luiz Velho, **Hermite interpolation of implicit surfaces with radial basis functions**. *Proc. of the Conference on Graphics, Patterns and Images – SIBGRAPI*, pages 1–8, Rio de Janeiro, Brazil, 2009. IEEE. (Cité page 28.)

[223] Benoit B. Mandelbrot, **The Fractal Geometry of Nature**. W. H. Freeman & Co Ltd, 1982. (Cité pages 35 et 43.)

[224] Benoit B. Mandelbrot, **The Science of Fractal Images**, chapitre Fractal Landscapes Without Creases and with Rivers, pages 243–260. Springer, 1 édition, 1988. (Cité page 36.)

[225] Benoit B. Mandelbrot et John W. Van Ness, **Fractional brownian motions, fractional noises and applications**. *SIAM Review – SIREV*, 10(4):422–437, 1968. (Cité page 33.)

[226] Nicolas Maréchal, **Génération de contenu graphique**. Thèse de doctorat, Université de Lyon, 2010. (Cité pages 10 et 56.)

[227] Nicolas Maréchal, Éric Guérin, Éric Galin, Stéphane Mérillou et Nicolas Mérillou, **Heat transfer simulation for modeling realistic winter sceneries**. *Computer Graphics Forum – CGF*, 29(2):449–458, 2010. (Cité pages 56 et 57.)

[228] Ivo Marák, Bedřich Beneš et Pavel Slavík, **Terrain erosion based on rewriting of matrices**. *Journal of WSCG – WSCG*, pages 341–351, 1997. (Cité page 50.)

[229] Iain Martin, Steve Parkes et Martin Dunstan, **Modeling cratered surfaces with real and synthetic terrain for testing planetary landers**. *Transaction on Aerospace and Electronic Systems – AESS*, 50(4):2916–2928, 2014. (Cité page 44.)

[230] Iain Martin, Steve Parkes, Martin Dunstan et Nick Rowell, **Asteroid modeling for testing spacecraft approach and landing**. *Computer Graphics and Applications – CG&A*, 34(4):52–62, 2014. (Cité page 44.)

[231] Gary A. Mastin, Peter A. Watterberg et John F. Mareda, **Fourier synthesis of ocean scenes**. *Computer Graphics and Applications – CG&A*, 7(3):16–23, 1987. (Cité page 38.)

[232] Wolfram MathWorld, **Cubic formula**. *Site Web* http://mathworld.wolfram.com/CubicFormula.html, 2015. (visité le 05/2015). (Cité page 77.)

[233] Wolfram MathWorld, **Point-quadratic distance**. *Site Web* http://mathworld.wolfram.com/Point-QuadraticDistance.html, 2015. (visité le 05/2015). (Cité page 77.)

[234] Colt McAnlis, BONFIRE Studios, SMU GuildHall et Microsoft Ensemble Studios, **Halo wars : The terrain of next gen**. *Site Web* http://www.gdcvault.com/play/1277/HALO-WARS-The-Terrain-of, 2011. (visité le 05/2015). (Cité page 17.)

[235] Morgan McGuire et Peter Sibley, **A heightfield on an isometric grid**. *Proc. of SIGGRAPH Technical Sketches and Applications – SIGGRAPH*, Los Angeles, USA, 2004. ACM SIGGRAPH. (Cité page 18.)

[236] Morgan McGuire et Peter Sibley, **A heightfield on an isometric grid**. Rapport technique, Brown University, 2005. (Cité page 18.)

[237] James McNeill, **Rapidly-exploring random trees**. *Site Web* http://playtechs.blogspot.fr/2008/11/rapidly-exploring-random-trees.html, 2008. (visité le 05/2015). (Cité page 162.)

[238] James McNeill, **Rrt with obstacles**. *Site Web* http://playtechs.blogspot.fr/2008/11/rrt-with-obstacles.html, 2008. (visité le 05/2015). (Cité page 162.)

[239] Xing Mei, Philippe Decaudin et Baogang Hu, **Fast hydraulic erosion simulation and visualization on GPU**. *Proc. of the Pacific Conference on Computer Graphics and Applications – PG*, pages 47–56, Maui, USA, 2007. IEEE. (Cité page 53.)

[240] Paul Merrell et Dinesh Manocha, **Example-based curve synthesis**. *Computers & Graphics – C&G*, 34(4):304–311, 2010. (Cité page 47.)

[241] Tian-De Miao, Qing-Song Mu et Sheng-Zhi Wu, **Computer simulation of aeolian sand ripples and dunes**. *Physics Letters A*, 288(1):16–22, 2001. (Cité page 55.)

[242] Elie Michel, **Génération procédurale de paysages à partir de cartes vectorielles**. Rapport de stage, ENS, 2014. (Cité page 49.)

[243] Elie Michel, Arnaud Emilien et Marie-Paule Cani, **Generation of folded terrains from simple vector maps**. *Short at Eurographics – EG*, Zürich, Switzerland, 2015. Eurographics Association. (Cité page 49.)

[244] Mike et Jeff, **explained xkcd**. *Site Web* http://www.explainxkcd.com/, 2009. (visité le 08/2015). (Cité page ix.)

[245] Gavin S. P. Miller, **The definition and rendering of terrain maps**. *Computer Graphics*, 20(4):39–48, 1986. (Cité page 36.)

[246] Mojang, **Minecraft**. *Jeu-vidéo* http://www.mobygames.com/game/minecraft, 2011. Microsoft. (Cité pages 17, 23, et 24.)

[247] Peter Molnar, **Geomorphology : Nature, nurture and landscape**. *Nature*, 426(6967):612–614, 2003. (Cité page 34.)

[248] Simon Monecke, **Implementing the paper "terrain generation using procedural models based on hydrology"**. Rapport de projet, Wedel University of Applied Sciences, 2014. (Cité page 163.)

[249] Marion Montaigne, **Tu mourras moins bête (mais tu mourras quand même)**. *Site Web* `http://tumourrasmoinsbete.blogspot.fr/`, 2008. (visité le 08/2015). (Cité page ix.)

[250] Randall Munroe, **xkcd**. *Site Web* `https://xkcd.com/`, 2005. (visité le 08/2015). (Cité page ix.)

[251] Kazunobu Muraoka et Norishige Chiba, **Visual simulation of snowfall, snow cover and snowmelt**. *Proc of the International Conference on Parallel and Distributed Systems – ICPADS*, pages 187–194, Iwate, Japan, 2000. IEEE. (Cité page 56.)

[252] Forest Kenton Musgrave, **Grid tracing : Fast ray tracing for height fields**. Rapport technique, Yale University, 1990. (Cité page 112.)

[253] Forest Kenton Musgrave, **Methods for Realistic Landscape Imaging**. Thèse de doctorat, Yale University, 1993. (Cité pages 34 et 43.)

[254] Forest Kenton Musgrave, **Procedural fractal terrains**. Rapport technique, Fractal-Worlds.com, 2000. (Cité pages 39 et 40.)

[255] Forest Kenton Musgrave, Craig E. Kolb et Robert S. Mace, **The synthesis and rendering of eroded fractal terrains**. *Computer Graphics*, 23(3):41–50, 1989. (Cité pages 34, 40, 49, 51, et 52.)

[256] Kenji Nagashima, **Computer generation of eroded valley and mountain terrains**. *The Visual Computer – TVC*, 13(9-10):456–464, 1998. (Cité page 52.)

[257] Mattia Natali, Endre M. Lidal, Julius Parulek, Ivan Viola et Daniel Patel, **Modeling terrains and subsurface geology**. *Proc. of Eurographics State Of The Art – EG STAR*, pages 155–173, Girona, Spain, 2013. Eurographics Association. (Cité page 11.)

[258] Mattia Natali, Julius Parulek et Daniel Patel, **Rapid modelling of interactive geological illustrations with faults and compaction**. *Proc. of the Spring Conference on Computer Graphics – SCCG*, Smolenice, Slovakia, 2014. Comenius University. (Cité pages 61 et 62.)

[259] Mattia Natali, Ivan Viola et Daniel Patel, **Rapid visualization of geological concepts**. *Proc. of the Conference on Graphics, Patterns and Images – SIBGRAPI*, pages 150–157, Ouro Petro, Brazil, 2012. IEEE. (Cité page 62.)

[260] B. Neidhold, M. Wacker et Olivier Deussen, **Interactive physically based fluid and erosion simulation**. *Proc. of the Eurographics Workshop on Natural Phenomena – EGWNP*, pages 25–32, Dublin, Ireland, 2005. Eurographics Association. (Cité page 52.)

[261] Fabrice Neyret, **Complexité Naturelle et Synthèse d'Images**. Habilitation à Diriger des Recherches, CNRS, 2001. (Cité page 9.)

[262] Fabrice NEYRET, **Créer, simuler, explorer des univers naturels sur ordinateur**. *Talk at the Journées de l'Association Française d'Informatique Graphique – AFIG*, Bordeaux, France, 2006. IRIT Presse. (Cité page 9.)

[263] Fabrice NEYRET et Nathalie PRAIZELIN, **Phenomenological simulation of brooks**. *Proc. of Eurographics Workshop on Computer Animation and Simulation – EGCAS*, pages 53–64, Manchester, UK, 2001. Springer. (Cité page 47.)

[264] Hiraku NISHIMORI et Noriyuki OUCHI, **Formation of ripple patterns and dunes by wind-blown sand**. *Physical review letters*, 71(1):197–201, 1993. (Cité page 55.)

[265] Tomoyuki NISHITA, Hiroshi IWASAKI, Yoshinori DOBASHI et Eihachiro NAKAMAE, **A modeling and rendering method for snow by using metaballs**. *Computer Graphics Forum – CGF*, 16(3):357–364, 1997. (Cité page 56.)

[266] Rockstar NORTH, **Grand theft auto v**. *Jeu-vidéo* `http://www.mobygames.com/game/grand-theft-auto-v`, 2013. Rockstar Games. (Cité page 4.)

[267] NOVALOGIC, **Comanche : Maximum overkill**. *Jeu-vidéo* `http://www.mobygames.com/game/dos/comanche-maximum-overkill`, 1992. NovaLogic. (Cité page 17.)

[268] NOVALOGIC, **Delta force**. *Jeu-vidéo* `http://www.mobygames.com/game/delta-force`, 1998. NovaLogic. (Cité page 17.)

[269] J.F. O'CALLAGHAN et D.M. MARK, **The extraction of drainage networks from digital elevation data**. *Computer Vision, Graphics, and Image Processing*, 28(3):323–344, 1984. (Cité pages 47 et 131.)

[270] Jacob OLSEN, **Realtime procedural terrain generation**. Rapport technique, University of Southern Denmark, 2004. (Cité pages 49, 50, et 51.)

[271] Teong Joo ONG, Ryan SAUNDERS, John KEYSER et John J. LEGGETT, **Terrain generation using genetic algorithms**. *Proc. of the Annual Conference on Genetic and Evolutionary Computation – GECCO*, pages 1463–1470, Washington D.C., USA, 2005. ACM. (Cité page 41.)

[272] Koichi ONOUE et Tomoyuki NISHITA, **A method for modeling and rendering dunes with wind-ripples**. *Proc. of the Pacific Conference on Computer Graphics and Applications – PG*, pages 427–428, Hong Kong, China, 2000. IEEE. (Cité page 55.)

[273] Koichi ONOUE et Tomoyuki NISHITA, **Virtual sandbox**. *Proc. of the Pacific Conference on Computer Graphics and Applications – PG*, pages 252–259, Canmore, Canada, 2003. IEEE. (Cité page 50.)

[274] Alexandrina ORZAN, Adrien BOUSSEAU, Holger WINNEMÖLLER, Pascal BARLA, Joëlle THOLLOT et David SALESIN, **Diffusion curves : A vector representation for smooth-shaded images**. *Transaction on Graphics – TOG*, 27(3):92 :1–82 :8, 2008. (Cité pages 28 et 62.)

[275] Ian PARBERRY, **Amortized noise**. *Journal of Computer Graphics Techniques – JCGT*, 3(2):31–47, 2014. (Cité page 39.)

[276] Ian Parberry, **Designer worlds : Procedural generation of infinite terrain from real-world elevation data**. *Journal of Computer Graphics Techniques – JCGT*, 3(1):74–85, 2014. (Cité pages 40, 59, et 60.)

[277] Ian Parberry, **Tobler's first law of geography, self similarity, and perlin noise : A large scale analysis of gradient distribution in southern utah with application to procedural terrain generation**. Rapport technique, University of North Texas, 2014. (Cité pages 40 et 60.)

[278] Ian Parberry, **Modeling real-world terrain with exponentially distributed noise**. *Journal of Computer Graphics Techniques – JCGT*, 4(2):1–9, 2015. (Cité pages 40 et 60.)

[279] Amit Patel, **Polygonal map generation for games**. *Site Web* `http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/`, 2010. (visité le 05/2015). (Cité pages 16, 44, et 45.)

[280] Amit Patel, **Curved paths**. *Site Web* `http://www.redblobgames.com/articles/curved-paths/`, 2013. (visité le 05/2015). (Cité page 80.)

[281] Amit Patel, **Hexagonal grids**. *Site Web* `http://www.redblobgames.com/grids/hexagons/`, 2013. (visité le 05/2015). (Cité page 18.)

[282] Amit Patel, **Noise functions and map generation**. *Site Web* `http://www.redblobgames.com/articles/noise/introduction.html`, 2013. (visité le 05/2015). (Cité page 38.)

[283] Daniel Patel, Christopher Giertsen, John Thurmond, John Gjelberg et Eduard Gröller, **The seismic analyzer : Interpreting and illustrating 2d seismic data**. *Transactions on Visualization and Computer Graphics – TVCG*, 14(6):1571–1578, 2008. (Cité page 65.)

[284] Daniel Patel, Christopher Giertsen, John Thurmond et Eduard Gröller, **Illustrative rendering of seismic data**. *Proc. of Vision, Modeling, and Visualization – VMV*, Saarbrücken, Germany, 2007. (Cité page 65.)

[285] PCGWiki, **Fractal river basins**. *Site Web* `pcg.wikidot.com/pcg-algorithm:fractal-river-basins`, 2011. (visité le 05/2015). (Cité page 47.)

[286] Darwyn R. Peachey, **Solid texturing of complex surfaces**. *Computer Graphics*, 19(3):279–286, 1985. (Cité page 39.)

[287] Vivi Kathrine Pedersen et David Lundbek Egholm, **Glaciations in response to climate variations preconditioned by evolving topography**. *Nature*, 493:206–210, 2013. (Cité pages 56 et 57.)

[288] Ken Perlin, **An image synthesizer**. *Computer Graphics*, 19(3):287–296, 1985. (Cité pages 10 et 39.)

[289] Ken Perlin, **Hypertexture**. *Computer Graphics*, 23(3):253–262, 1989. (Cité page 39.)

[290] Ken PERLIN, **Noise hardware**. *Proc. of SIGGRAPH Course Notes – SIGGRAPH*, Los Angeles, USA, 2001. ACM SIGGRAPH. (Cité page 39.)

[291] Ken PERLIN, **Improving noise**. *Transaction on Graphics – TOG*, 21(3):681–682, 2002. (Cité page 39.)

[292] Ken PERLIN et Fabrice NEYRET, **Flow noise**. *Proc. of SIGGRAPH Technical Sketches and Applications – SIGGRAPH*, Los Angeles, USA, 2001. ACM SIGGRAPH. (Cité page 39.)

[293] Hélène PERRIER, **Génération de niveaux de détails continus**. Mémoire de Master, Université Claude Bernard Lyon 1, 2014. (Cité page 106.)

[294] Thomas K. PEUCKER et Nicholas CHRISMAN, **Cartographic data structures**. *The American Cartographer*, 2(1):55–69, 1975. (Cité page 20.)

[295] Thomas K. PEUCKER, Robert J. FOWLER, James J. LITTLE et David M. MARK, **The triangulated irregular network**. *Proc. of Digital Terrain Models Symboposium – DTM*, St. Louis, USA, 1978. American Society of Photogrammetry. (Cité page 20.)

[296] Alexandre PEYRAT, Olivier TERRAZ, Stéphane MÉRILLOU, Éric GALIN et Djamchid GHAZANFARPOUR, **Generating large-scale details : Altering soil surface and structure with tracks**. *Proc. of Virtual Reality Interactions and Physical Simulations – VRIPHYS*, Lyon, France, 2011. Eurographics Association. (Cité pages 22, 23, et 50.)

[297] Adrien PEYTAVIE, **Génération Procédurale de Monde**. Thèse de doctorat, Université de Lyon, 2010. (Cité pages 10 et 24.)

[298] Adrien PEYTAVIE, Éric GALIN, Jérome GROSJEAN et Stéphane MÉRILLOU, **Modélisation de terrains complexes 3d**. *Proc. of the Journées de l'Association Française d'Informatique Graphique – AFIG*, Toulouse, France, 2008. IRIT Presse. (Cité page 24.)

[299] Adrien PEYTAVIE, Éric GALIN, Stephane MÉRILLOU et Jérôme GROSJEAN, **Arches : a framework for modeling complex terrains**. *Computer Graphics Forum – CGF*, 28(2):457–467, 2009. (Cité pages 24, 97, et 101.)

[300] Nico PIETRONI, Paolo CIGNONI, Miguel A. OTADUY et Roberto SCOPIGNO, **Solid-texture synthesis : a survey**. *Computer Graphics and Applications – CG&A*, 30(4):74–89, 2010. (Cité page 59.)

[301] Sören PIRK, Till NIESE, Torsten HÄDRICH, Bedrich BENES et Oliver DEUSSEN, **Windy trees : Computing stress response for developmental tree models**. *Transaction on Graphics – TOG*, 33(6):204 :1–204 :11, 2014. (Cité page 55.)

[302] PIXOLOGIC, **Zbrush**. *Logiciel* `http://pixologic.com/zbrush/`, 1999. (Cité pages 23 et 167.)

[303] Fabio POLICARPO et Manuel M. OLIVEIRA, **Relaxed cone stepping for relief mapping**. *GPU Gems 3*, volume 3 de *GPU Gems*, pages 409–428. Addison-Wesley Professional, 2008. (Cité page 113.)

[304] Pomax, **A primer on bézier curves**. *Site Web* http://pomax.github.io/bezierinfo/, 2013. (visité le 05/2015). (Cité page 78.)

[305] Joachim Pouderoux, Jean-Christophe Gonzato, Ireneusz Tobor et Pascal Guitton, **Adaptive hierarchical rbf interpolation for creating smooth digital elevation models**. *Proc. of the International Workshop on Geographic Information Systems*, Washington D.C., USA, 2004. ACM. (Cité pages 28 et 29.)

[306] Simon Premože, William B. Thompson et Peter Shirley, **Geospecific rendering of alpine terrain**. *Proc. of the Workshop on Rendering Techniques – EGWR*, Granada, Spain, 1999. Eurographics Association. (Cité pages 56 et 57.)

[307] CD Projekt, **The witcher 3**. *Jeu-vidéo* http://en.wikipedia.org/wiki/The_Witcher_3:_Wild_Hunt, 2015. CD Projekt. (Cité page 2.)

[308] Przemyslaw Prusinkiewicz et Marc Hammel, **A fractal model of mountains with rivers**. *Proc. of Graphics Interface – GI*, pages 174–180, Toronto, Canada, 1993. Canadian Information Processing Society. (Cité pages 36 et 45.)

[309] Anna Puig-Centelles, Peter A. C. Varley, Oscar Ripolles et Miguel Chover, **Automatic terrain generation with a sketching tool**. *Multimedia Tools and Applications*, 70(3):1957–1986, 2014. (Cité page 61.)

[310] Alex Pytel et Stephen Mann, **Self-organized approach to modeling hydraulic erosion features**. *Poster at Graphics Interface – GI*, 2011. (Cité pages 50 et 52.)

[311] Alex Pytel et Stephen Mann, **Generalized maps for erosion and sedimentation simulation**. *Computers & Graphics – C&G*, 37(4):280–292, 2013. (Cité pages 50 et 52.)

[312] Alex Pytel et Stephen Mann, **Procedural modeling of cave-like channels**. *Journal of Computer Graphics Techniques – JCGT*, 4(2):10–29, 2015. (Cité page 46.)

[313] Huamin Qu, Feng Qiu, Nan Zhang, Arie Kaufman et Minge Wan, **Ray tracing height fields**. *Proc. of Computer Graphics International – CGI*, pages 202–207, Tokyo, Japan, 2003. IEEE. (Cité page 112.)

[314] Iñigo Quílez, **Advanced perlin noise**. *Site Web* http://www.iquilezles.org/www/articles/morenoise/morenoise.htm, 2008. (visité le 05/2015). (Cité pages 39 et 40.)

[315] Iñigo Quílez, **Inverse bilinear interpolation**. *Site Web* http://www.iquilezles.org/www/articles/ibilinear/ibilinear.htm, 2010. (visité le 05/2015). (Cité page 81.)

[316] Iñigo Quílez, **One of those weekends**. *Site Web* http://www.iquilezles.org/blog/?p=2082, 2013. (visité le 05/2015). (Cité page 6.)

[317] Iñigo Quílez, **Voronoise**. *Site Web* http://www.iquilezles.org/www/articles/voronoise/voronoise.htm, 2014. (visité le 05/2015). (Cité page 39.)

[318] William L. Raffe, Fabio Zambetta et Xiaodong Li, **Evolving patch-based terrains for use in video games**. *Proc. of the Annual Conference on Genetic and Evolutionary Computation – GECCO*, pages 363–370, Dublin, Ireland, 2011. ACM. (Cité pages 41 et 42.)

[319] William L. Raffe, Fabio Zambetta et Xiaodong Li, **A survey of procedural terrain generation techniques using evolutionary algorithms**. *Proc. of the Congress of Evolutionary Computation – CEC*, pages 2090–2097, Brisbane, Australia, 2012. IEEE. (Cité page 41.)

[320] S. Reed, O. Kreylos, S. Hsi, L. Kellogg, G. Schladow, M.B. Yikilmaz, H. Segale, J. Silverman, S. Yalowitz et E. Sato, **Augmented reality sandbox**. *Site Web* http://idav.ucdavis.edu/~okreylos/ResDev/SARndbox/, 2014. (visité le 05/2015). (Cité page 168.)

[321] Matthieu Richard, **Simulation visuelle d'avalanches poudreuses**. Mémoire de Master, Université Joseph Fourier, 2003. (Cité page 56.)

[322] Abigail Martínez Rivas et Luis Gerardo De La Fraga, **Terrain reconstruction from contour maps**. *Proc. of the International Workshop on Visualization and Animation of Landscape*, pages 167–175, Kunming, China, 2005. (Cité page 19.)

[323] Tiphaine Rivière, **Le bureau 14 de la Sorbonne / carnets de thèse**. *Site Web* https://lebureau14delasorbonne.wordpress.com/, 2010. (visité le 08/2015). (Cité page ix.)

[324] Frank Rochet, **Simulation réaliste de ruisseaux en temps réel**. Mémoire de Master, Université Joseph Fourier de Grenoble, 2005. (Cité page 47.)

[325] David Rosgen, **The key to the rosgen classification of natural rivers**. *Site Web* http://www.fgmorph.com/images/04/04_16_01.jpg, 2001. (visité le 05/2015). (Cité page 134.)

[326] David L. Rosgen, **A classification of natural rivers**. *Catena*, 22(3):169–199, 1994. (Cité pages 133, 137, et 149.)

[327] David L. Rosgen et Hilton Lee Silvey, **Applied River Morphology**, volume 1481. Wildland Hydrology, Pagosa Springs, Colorado, 1996. (Cité page 133.)

[328] Pascale Roudier, Bernard Peroche et Michel Perrin, **Landscapes synthesis achieved through erosion and deposition process simulation**. *Computer Graphics Forum – CGF*, 12(3):375–383, 1993. (Cité pages 49, 51, et 91.)

[329] Gilles Roussel, **Bouletcorp**. *Site Web* http://www.bouletcorp.com/, 2004. (visité le 08/2015). (Cité page ix.)

[330] Witawat Rungjiratananon, Zoltan Szego, Yoshihiro Kanamori et Tomoyuki Nishita, **Real-time animation of sand-water interaction**. *Computer Graphics Forum – CGF*, 27(7):1887–1893, 2008. (Cité page 48.)

[331] Brennan Rusnell, David Mould et Mark G. Eramian, **Feature-rich distance-based terrain synthesis**. *The Visual Computer – TVC*, 25(5-7):573–579, 2009. (Cité page 61.)

[332] Ryan, **Strengths and weaknesses of procedural generation for games**. *Site Web* `http://procedurallygenerated.com/?p=111`, 2012. (visité le 05/2015). (Cité page 27.)

[333] Geiss Ryan, **Generating complex procedural terrains using the gpu**. *GPU Gems 3*, volume 3 de *GPU Gems*, pages 7–37. Addison-Wesley Professional, 2007. (Cité page 40.)

[334] Georgios Sakas, **Modeling and animating turbulent gaseous phenomena using spectral synthesis**. *The Visual Computer – TVC*, 9(4):200–212, 1993. (Cité page 38.)

[335] Aitor Santamaría-Ibirika, Xabier Cantero, Mikel Salazar, Jaime Devesa et Pablo G. Bringas, **Volumetric virtual worlds with layered terrain generation**. *Proc. of the International Conference on Cyberworlds*, pages 20–27, Yokohama, Japan, 2013. IEEE. (Cité page 24.)

[336] Aitor Santamaría-Ibirika, Xabier Cantero, Mikel Salazar, Jaime Devesa, Igor Santos, Sergio Huerta et Pablo G. Bringas, **Procedural approach to volumetric terrain generation**. *The Visual Computer – TVC*, 30(9):997–1007, 2013. (Cité page 24.)

[337] V. B. Sapozhnikov et V. I. Nikora, **Simple computer model of a fractal river network with fractal individual watercourses**. *Journal of Physics A : Mathematical and General*, 26(15), 1993. (Cité page 46.)

[338] Ryan L. Saunders, **Terrainosaurus : Realistic terrain synthesis using genetic algorithms**. Mémoire de Master, Texas A&M University, 2006. (Cité page 41.)

[339] Dietmar Saupe, **The Science of Fractal Images**, chapitre Algorithms for random fractals, pages 71–136. Springer, 1 édition, 1988. (Cité page 37.)

[340] Dietmar Saupe, **Point evaluation of multi-variable random fractals**. *Visualisierung in Mathematik und Naturwissenschaften*, pages 114–126. Springer, 1989. (Cité page 40.)

[341] Jens Schneider, Tobias Boldte et Rüdiger Westermann, **Real-time editing, synthesis, and rendering of infinite landscapes on GPUs**. *Proc. of Vision, Modeling, and Visualization – VMV*, Aachen, Germany, 2006. (Cité page 40.)

[342] Donald Shepard, **A two-dimensional interpolation function for irregularly-spaced data**. *Proc. of the ACM National Conference*, 1968. (Cité page 31.)

[343] Byeong-Seok Shin et Hoe Sang Jung, **Contour-based terrain model reconstruction using distance information**. *Proc. of the International Conference on Computational Science and Its Applications — ICCSA*, pages 1177–1186, Singapore, Singapore, 2005. Springer. (Cité page 19.)

[344] Byeong-Seok Shin et Hoe Sang Jung, **Fast reconstruction of 3d terrain model from contour lines on 2d maps**. *Proc. of the Asian Simulation Conference — AsiaSim*, pages 230–239, Jeju Island, Korea, 2005. Springer. (Cité page 19.)

[345] Robin Sibson, **A brief description of natural neighbor interpolation**, chapitre 2, pages 21–36. John Wiley and Sons Ltd., 1981. (Cité page 32.)

[346] Green Simon, **Implementing improved perlin noise**. *GPU Gems 2*, volume 2 de *GPU Gems*, pages 409–416. Addison-Wesley Professional, 2005. (Cité page 40.)

[347] Branislav Síleš, **Atomontage engine**. *Logiciel* http://www.atomontage.com/, 2015. (Cité page 24.)

[348] Ruben M. Smelik, Tim Tutenel, Rafael Bidarra et Bedřich Beneš, **A survey on procedural modelling for virtual worlds**. *Computer Graphics Forum – CGF*, 33(6):31–50, 2014. (Cité page 11.)

[349] Ruben M. Smelik, Tim Tutenel, Klaas Jan De Kraker et Rafael Bidarra, **Declarative terrain modeling for military training games**. *International Journal of Computer Games Technology – IJCGT*, 2010. (Cité page 63.)

[350] Ruben M. Smelik, Tim Tutenel, Klaas Jan De Kraker et Rafael Bidarra, **Integrating procedural generation and manual editing of virtual worlds**. *Proc. of the Workshop on Procedural Content Generation in Games – PCGames*, Monterey, USA, 2010. ACM. (Cité page 63.)

[351] Ruben M. Smelik, Tim Tutenel, Klaas Jan De Kraker et Rafael Bidarra, **Integrating procedural generation and manual editing of virtual worlds**. *Short at Eurographics – EG*, Norrköping, Sweden, 2010. (Cité page 63.)

[352] Ruben M. Smelik, Tim Tutenel, Klaas Jan De Kraker et Rafael Bidarra, **A declarative approach to procedural modeling of virtual worlds**. *Computers & Graphics – C&G*, 35(2): 352–363, 2011. (Cité pages 63 et 64.)

[353] Ruben M. Smelik, Tim Tutenel, Klaas Jan De Kraker et Rafael Bidarra, **Semantic constraints for procedural modelling of virtual worlds**. *Proc. of the Workshop on Procedural Content Generation in Games – PCGames*, Bordeaux, France, 2011. ACM. (Cité page 63.)

[354] Ruben Michaël Smelik, **A Declarative Approach to Procedural Generation of Virtual Worlds**. Thèse de doctorat, Technische Universiteit Delf, 2011. (Cité page 63.)

[355] Side Effects Software, **Houdini**. *Logiciel* http://www.sidefx.com/, 1999. (Cité pages 6 et 167.)

[356] Carlo H. Séquin, Kiha Lee et Jane Yen, **Fair, G^2- and C^2- continuous circle splines for the interpolation of sparse data points**. *Computer-Aided Design – CAD*, 37(2):201–211, 2005. (Cité page 78.)

[357] Szymon Stachniak et Wolfgang Stuerzlinger, **An algorithm for automated fractal terrain deformation**. *Proc. of the International Conference on Computer Graphics and Artificial Intelligence – 3IA*, pages 64–76, Limoges, France, 2005. (Cité pages 64 et 65.)

[358] Arthur Newell Strahler, **Hypsometric (area-altitude) analysis of erosional topology**. *Geological society of America bulletin*, 63(11):1117–1142, 1952. (Cité page 131.)

[359] Amplitude Studios, **Endless legend**. *Jeu-vidéo* http://en.wikipedia.org/wiki/Endless_Legend, 2014. Iceberg Interactive. (Cité page 17.)

B

[360] I-Novae Studios, **I-novae engine**. *Logiciel* https://inovaestudios.com/, 2014. (Cité page 5.)

[361] Robert W. Sumner, James F. O'Brien et Jessica K. Hodgins, **Animating sand, mud and snow**. *Proc. of Graphics Interface – GI*, pages 125–132, Vancouver, Canada, 1998. Canadian Information Processing Society. (Cité page 50.)

[362] Robert W. Sumner, James F. O'Brien et Jessica K. Hodgins, **Animating sand, mud and snow**. *Computer Graphics Forum – CGF*, 18(1):17–26, 1999. (Cité page 50.)

[363] Ben Sutherland et John Keyser, **Particle-based enhancement of terrain data**. *Poster at SIGGRAPH*, 2006. (Cité page 52.)

[364] Sven/Fulcrum, **Rendering mandelbox fractals faster with cone marching**. *Site Web* http://www.fulcrum-demo.org/wp-content/uploads/2012/04/Cone_Marching_Mandelbox_by_Seven_Fulcrum_LongVersion.pdf, 2012. (visité le 05/2015). (Cité page 113.)

[365] Ondřej Šťava, Bedřich Beneš, Matthew Brisbin et Jaroslav Křivánek, **Interactive terrain modeling using hydraulic erosion**. *Proc. of Symposium on Computer Animation – SCA*, pages 201–210, Dublin, Ireland, 2008. Eurographics Association. (Cité pages 10 et 53.)

[366] Richard Szeliski et Demetri Terzopoulos, **From splines to fractals**. *Computer Graphics*, 23(3):51–60, 1989. (Cité page 36.)

[367] Kenshi Takayama, Olga Sorkine, Andrew Nealen et Takeo Igarashi, **Volumetric modeling with diffusion surfaces**. *Transaction on Graphics – TOG*, 29(6):180 :1–180 :8, 2010. (Cité pages 29 et 62.)

[368] David G. Tarboton, **A new method for the determination of flow directions and upslope areas in grid digital elevation models**. *Water Resources Research*, 33(2):309–319, 1997. (Cité page 131.)

[369] Flora Ponjou Tasse, **Distributed texture-based terrain synthesis**. Mémoire de Master, University of Cape Town, 2011. (Cité pages 60 et 61.)

[370] Flora Ponjou Tasse, Arnaud Emilien, Marie-Paule Cani, Stefanie Hahmann et Adrien Bernhardt, **First person sketch-based terrain editing**. *Proc. of Graphics Interface – GI*, pages 217–224, Montreal, Canada, 2014. Canadian Information Processing Society. (Cité page 61.)

[371] Flora Ponjou Tasse, Arnaud Emilien, Marie-Paule Cani, Stefanie Hahmann et Neil Dodgson, **Feature-based terrain editing from complex sketches**. *Computers & Graphics – C&G*, 45:101–115, 2014. (Cité page 61.)

[372] Flora Ponjou Tasse, James E. Gain et Patrick Marais, **Enhanced texture-based terrain synthesis on graphics hardware**. *Computer Graphics Forum – CGF*, 31(6):1959–1972, 2012. (Cité pages 10, 60, et 61.)

[373] Antoine TAVENEAUX, **Puissance logique et calculatoire de l'aléa algorithmique**. Thèse de doctorat, Université Paris 7 Diderot, 2013. (Cité page x.)

[374] Rgba & TBC, **Elevated**. *Site Web* `http://www.pouet.net/prod.php?which=52938`, 2009. (visité le 05/2015). (Cité page 5.)

[375] Soon Tee TEOH, **River and coastal action in automatic terrain generation**. *Proc. of Computer Graphics and Virtual Reality – CGVR*, pages 3–9, Las Vegas, USA, 2008. CSREA Press. (Cité pages 45 et 151.)

[376] Soon Tee TEOH, **RiverLand : An efficient procedural modeling system for creating realistic-looking terrains**. *Proc. of the International Symposium on Advances in Visual Computing – ISVC*, pages 468–479, Las Vegas, USA, 2009. Springer. (Cité pages 45 et 151.)

[377] Art TEVS, Ivo IHRKE et Hans-Peter SEIDEL, **Maximum mipmaps for fast, accurate, and scalable dynamic height field rendering**. *Proc. of the Symposium on Interactive 3D Graphics and Games – I3D*, pages 183–190, Redwood City, USA, 2008. ACM. (Cité page 112.)

[378] David THIBAULT et Christopher M. GOLD, **Terrain reconstruction from contours by skeleton construction**. *GeoInformatica*, 4(4):349–373, 2000. (Cité page 19.)

[379] Rastislav TISOVČÍK, **Generation and visualization of terrain in virtual environment**. Rapport de Licence, Masaryk University, 2012. (Cité page 43.)

[380] Julian TOGELIUS, Georgios N. YANNAKAKIS, Kenneth O. STANLEY et Cameron BROWNE, **Search-based procedural content generation : A taxonomy and survey**. *Transactions on Computational Intelligence and AI in Games – T-CIAIG*, 3(3):172–186, 2011. (Cité page 11.)

[381] Éric TOSAN, **Modélisation géométrique : de l'approche classique à l'approche fractale**. Habilitation à Diriger des Recherches, Université Claude Bernard Lyon 1, 2005. (Cité page 36.)

[382] Yusuke TSUDA, Yonghao YUE, Yoshinori DOBASHI et Tomoyuki NISHITA, **Visual simulation of mixed-motion avalanches with interactions between snow layers**. *The Visual Computer – TVC*, 26(6-8):883–891, 2010. (Cité page 56.)

[383] Shih-Chun TU, Chun-Yen HUANG et Wen-Kai TAI, **Terrain synthesis based on microscopic terrain feature**. *Proc. of the International Conference on Technologies for E-Learning and Digital Entertainment*, pages 644–655, Nanjing, China, 2005. Springer. (Cité page 59.)

[384] Gregory E. TUCKER, Stephen T. LANCASTER, Nicole M. GASPARINI, Rafael L. BRAS et Scott M. RYBARCZYK, **An object-oriented framework for distributed hydrologic and geomorphic modeling using triangulated irregular networks**. *Computers & Geosciences*, 27(8):959–973, 2001. (Cité page 20.)

[385] Luther A. TYCHONIEVICH et Michael D. JONES, **Delaunay deformable mesh for the weathering and erosion of 3d terrain**. *The Visual Computer – TVC*, 26(12):1485–1495, 2010. (Cité pages 53, 57, et 58.)

[386] Rodolphe Vaillant, **Recipe for implicit surface reconstruction with hrbf**. *Site Web* `http://rodolphe-vaillant.fr/?e=12`, 2013. (visité le 05/2015). (Cité page 28.)

[387] Wim van Eck et Maarten H. Lamers, **Evolving game terrains through living organisms**. *Talk at the Future of Procedural Content*, 2014. `http://bit.ly/1zWc5Wu`. (Cité page 44.)

[388] Wim van Eck et Maarten H. Lamers, **Evolving game terrains through living organisms**. *Proc. of the International Conference on Evolutionary and Biologically Inspired Music, Sound, Art and Design – EvoMUSART*, pages 1–12, Copenhagen, Denmark, 2015. Springer. (Cité page 44.)

[389] Frans van Hoesel, **Tiled directional flow**. *Poster at SIGGRAPH*, 2011. (Cité pages 47 et 48.)

[390] Joost van Pabst van Lawick et Hans Jense, **Dynamic terrain generation based on multifractal techniques**. *High Performance Computing for Computer Graphics and Visualisation*, pages 186–203, 1995. (Cité page 40.)

[391] Jarke J. Van Wijk, **Spot noise texture synthesis for data visualization**. *Computer Graphics*, 25(4):309–318, 1991. (Cité page 39.)

[392] Juraj Vanek, Bedřich Beneš, Adam Herout et Ondřej Šťava, **Large-scale physics-based terrain editing using adaptive tiles on the GPU**. *Computer Graphics and Applications – CG&A*, 31(6):35–44, 2011. (Cité page 53.)

[393] Peter A. C. Varley, Miguel Chover et Anna Puig-Centelles, **Sketching islands for a game environment**. *Proc. of European Conference on Visual Media Production – CVMP*, pages 1–10, London, U.K., 2008. (Cité page 61.)

[394] Lauri Viitanen, **Physically based terrain generation : Procedural heightmap generation using plate tectonics**. Rapport de Licence, Helsinki Metropolia University of Applied Sciences, 2012. (Cité pages 48 et 49.)

[395] Ulysse Vimont, **Génération de terrains par bassins versants hiérarchiques**. Mémoire de Master, CPE Lyon, 2013. (Cité pages 47, 131, 138, 158, 161, et 163.)

[396] Interactive Data Visualization, **Speedtree**. *Logiciel* `http://www.speedtree.com/`, 2000. (Cité page 6.)

[397] Emilio Vital Brazil, Ives Macêdo, Mario Costa Sousa, Luiz Henrique de Figueiredo et Luiz Velho, **Sketching variational hermite-rbf implicits**. *Proc. of the International Symposium on Sketch-Based Interfaces and Modeling – SBIM*, pages 1–8, Annecy, France, 2010. Eurographics Association. (Cité pages 28 et 29.)

[398] Enrique R. Vivoni, Valeri Y. Ivanov, Rafael L. Bras et Dara Entekhabi, **Generation of triangulated irregular networks based on hydrological similarity**. *Journal of Hydrologic Engineering*, 9(4):288–302, 2004. (Cité page 20.)

[399] Alex Vlachos, **Water flow in portal 2**. *Presentation at SIGGRAPH*, 2010. (Cité page 47.)

[400] Richard F. Voss, **Random fractal forgeries**. *Fundamental Algorithms for Computer Graphics*, volume 17, pages 805–835. Springer, 1991. (Cité page 43.)

[401] Paul Walsh et Prasad Gade, **Terrain generation using an interactive genetic algorithm**. *Proc. of the Congress on Evolutionary Computation – CEC*, pages 1–7, Barcelona, Spain, 2010. IEEE. (Cité page 42.)

[402] Lvdi Wang, Yizhou Yu, Kun Zhou et Baining Guo, **Multiscale vector volumes**. *Transaction on Graphics – TOG*, 30(6):167 :1–167 :8, 2011. (Cité pages 28 et 29.)

[403] Ning Wang et Bao-Gang Hu, **Real-time simulation of aeolian sand movement and sand ripple evolution : A method based on the physics of blown sand**. *Journal of Computer Science and Technology*, 27(1):135–146, 2012. (Cité pages 55 et 56.)

[404] Warhorse, **Kingdom come : Deliverance**. *Jeu-vidéo* http://en.wikipedia.org/wiki/Kingdom_Come:_Deliverance, 2015. Warhorse. (Cité page 3.)

[405] Nayuko Watanabe, **A sketching interface for terrain modeling**. Rapport de Licence, University of Tokyo, 2004. (Cité page 61.)

[406] Nayuko Watanabe et Takeo Igarashi, **A sketching interface for terrain modeling**. *Poster at SIGGRAPH*, 2004. (Cité page 61.)

[407] Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra et Greg Turk, **State of the art in example-based texture synthesis**. *Proc. of Eurographics State Of The Art – EG STAR*, pages 93–117, Munich, Germany, 2009. Eurographics Association. (Cité page 59.)

[408] John C. Whelan et Mahes Visvalingam, **Formulated silhouettes for sketching terrain**. *Proc. of Theory and Practice of Computer Graphics – TP.CG*, pages 90–96, Birmingham, UK, 2003. IEEE. (Cité page 60.)

[409] Mattias Widmark, **Terrain in battlefield 3 : A modern, complete and scalable system**. *Site Web* http://www.frostbite.com/wp-content/uploads/2013/05/GDC12_Terrain_in_Battlefield3.pdf, 2013. (visité le 05/2015). (Cité page 5.)

[410] T. A. Witten Jr et Leonard M. Sander, **Diffusion-limited aggregation, a kinetic critical phenomenon**. *Physical review letters*, 47(19), 1981. (Cité page 162.)

[411] Christopher Wojtan, Mark Carlson, Peter Mucha et Greg Turk, **Animating corrosion and erosion**. *Proc. of the Eurographics Workshop on Natural Phenomena – EGWNP*, pages 15–22, Prague, Czech Republic, 2007. Eurographics Association. (Cité pages 52, 53, et 58.)

[412] Joe Wolfe, **How to write a PhD thesis**. *Site Web* http://newt.phys.unsw.edu.au/~jw/thesis.html#coda, 1996. (visité le 08/2015). (Cité page x.)

[413] Steven Worley, **A cellular texture basis function**. *Proc. of SIGGRAPH Conference – SIGGRAPH*, pages 291–294, New Orleans, USA, 1996. ACM. (Cité page 39.)

[414] Burkhard Wünsche, Daniel Keymer et Robert Amor, **Sketch, click, plug and play : Accelerated design of virtual environments by integrating multimedia and sketch content into game engines**. *Proc. of Conference on Human-Computer Interaction of the ACM SIGCHI NZ – CHINZ*, pages 33–40, Auckland, New Zealand, 2010. ACM. (Cité page 61.)

[415] Brian Wyvill, Andrew Guy et Éric Galin, **Extending the CSG tree - warping, blending and boolean operations in an implicit surface modeling system**. *Computer Graphics Forum – CGF*, 18(2):149–158, 1999. (Cité pages 28, 74, 75, et 118.)

[416] Xin Xie et Burkhard C. Wünsche, **Efficient contour line labelling for terrain modelling**. *Proc. of the Australasian Computer Science Conference – ACSC*, pages 33–42, Brisbane, Australia, 2010. (Cité page 19.)

[417] Yafaray, **Lighting methods**. *Site Web* `http://www.yafaray.org/documentation/userguide/lightingmethods`, 2009. (visité le 05/2015). (Cité page 111.)

[418] Qizhi Yu, Fabrice Neyret, Eric Bruneton et Nicolas Holzschuch, **Scalable real-time animation of rivers**. *Computer Graphics Forum – CGF*, 28(2):239–248, 2009. (Cité page 47.)

[419] Simon Zaaijer, **GPU based generation and real-time rendering of semi-procedural terrain using features**. Mémoire de Master, Leiden University, 2013. (Cité page 40.)

[420] Zhiyi Zhang, Kouichi Konno et Yoshimasa Tokuyama, **3d terrain reconstruction based on contours**. *Proc. of the International Conference on Computer Aided Design and Computer Graphics – CAD/CG*, pages 325–330, Hong Kong, China, 2005. IEEE. (Cité page 19.)

[421] Howard Zhou, Jie Sun, Greg Turk et James M. Rehg, **Terrain synthesis from digital elevation models**. *Transactions on Visualization and Computer Graphics – TVCG*, 13(4):834–848, 2007. (Cité pages 59 et 61.)

[422] Chen Zhuo, Zhao Yanqing et Yang Chongjun, **Real-time contour map reconstruction with 3d terrain on modern graphics hardware**. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 110, 2009. (Cité page 19.)