Université Claude Bernard Lyon 1 – UFR informatique **École doctorale InfoMath**

CONTRÔLE DANS LA GÉNÉRATION DE FORMES NATURELLES

THÈSE

présentée et soutenue publiquement le 13 Septembre 2010

pour l'obtention du

Doctorat de l'université Claude Bernard – Lyon 1

(spécialité informatique)

par

Houssam HNAIDI

Composition du jury

Rapporteurs :

M^{me} Dominique FAUDOT	Professeur, Université de Bourgogne
M. Stéphane MÉRILLOU	Maître de conférences, Université de Limoges

Examinateurs :

M. Laurent LUCAS	Professeur, Université de Reims
M. Jean SEQUEIRA	Professeur, Université de Luminy

Directeurs :

M. Samir AKKOUCHE	Professeur, Université Claude Bernard Lyon1
M. ric GUÉRIN	Maître de conférences, Université Claude Bernard Lyon1

Laboratoire d'Informatique en Image et Système d'Information — UMR 5205

Remerciements

Je tiens tout d'abord à remercier Samir Akkouche et Éric Guérin, mes encadrants, qui m'ont permis d'effectuer cette thèse. Je remercie également Dominique Faudot et Stéphane Mérillou d'avoir accepté d'être rapporteurs et d'avoir pris le temps de lire en détail ce manuscrit malgré la quantité de fautes d'orthographe. Je remercie aussi Laurent Lucas et Jean Sequeira qui ont accepté d'être examinateurs.

Merci à tous les membres du LIRIS, avec eux j'ai passé des agréables moments dans ce laboratoire. Merci à mes co-bureaux de la TD6 qui m'ont supporté durant ces quatre ans. Un grand merci à tous mes amis qui m'ont supporté durant mon séjour en France.

Je tiens notamment à remercier toute ma famille : mes parents, mes frères et mes sœurs, qui m'ont toujours soutenu malgré la grande distance nous séparant.

Je tiens particulièrement à remercier mon épouse Ghezlan pour tout son soutien moral qui me donne le courage de continuer à avancer. Sans oublier de remercier mon petit fils Yani qui a enrichi ma vie.

Table des matières

Introduction

La génération de formes naturelles est un domaine de recherche important dans l'informatique graphique. Depuis déjà longtemps les chercheurs s'y sont intéressés dans des domaines d'application allant du loisir numérique aux effets spécieux de l'industrie cinématographique. La génération de formes naturelles concerne une grande variété d'objets parmi laquelle nous pouvons citer les caractéristiques de terrain (montagne, rivière, lac, *etc*), les arbres, les plantes, les feuilles, les cailloux, *etc*. Chacun de ces objets représente un domaine vaste de recherche. Un grand défi se pose avec la génération de tels objets : le contrôle de leurs formes. Pour cette raison de nombreuses méthodes ont été introduites afin de contrôler la forme des objets naturels tout en préservant leur apparence réaliste. Les techniques de génération sont très efficaces pour créer rapidement et automatiquement des milliers d'objets naturels, et les techniques de contrôle de forme sont un moyen efficace pour contrôler l'apparence des objets. Ainsi la fusion des techniques de génération et des techniques de contrôle est nécessaire, mais le contrôle de la forme des objets naturels devient beaucoup plus difficile quand le but est de contrôler le processus de génération de multiples objets plutôt que la forme d'un objet unique.

Durant une quarantaine d'années, de nombreuses méthodes (Chapitre 1) ont été proposées utilisant diverses techniques (esquisses, basée exemple, *etc*) afin de contrôler la forme générée des objets naturels en gardant leur aspect naturel. Par exemple, plusieurs méthodes permettent de contrôler la génération de terrain en introduisant des contraintes au modèle de génération, ou bien en partant d'un exemple réel (image de terrain, carte topographique,*etc*) ou d'esquisses tracées par l'utilisateur. Aussi, Dans le domaine de la génération de plantes, des méthodes sont introduites pour adapter, par exemple, le modèle L-système à un ensemble d'esquisses ou à une image de plante.

Les techniques de modélisation de formes naturelles peuvent être classées en trois catégories : la génération procédurale, les méthodes d'édition et la simulation. Les méthodes de génération procédurale permettent de créer un nombre important d'objets naturels en utilisant un ensemble de paramètres et une procédure de génération : il suffit de changer un seul paramètre pour générer un nouvel objet. Ces méthodes peuvent générer des ter-

rains, des forêts, des arbres, *etc.* Les méthodes d'édition présentent plus de contrôle sur la forme de l'objet. Ces méthodes nécessitent une part importante dans l'intervention de l'utilisateur ce qui les rend fastidieuses surtout si la création d'une variété d'objets est cherchée. Malgré cet inconvénient, ces méthodes, et surtout celles qui se basent sur des esquisses ou des exemples, sont les plus utilisées pour contrôler la génération des objets naturels. La dernière catégorie des méthodes essaie de simuler les phénomènes naturels en employant des lois physiques, chimiques ou biologiques afin d'augmenter le réalisme de l'objet traité. Ces méthodes, de par leur coût élevé, ne présentent qu'un faible contrôle sur la forme de l'objet.

Actuellement, de nombreuses méthodes existent pour générer un terrain, mais peu parmi elles le génèrent selon ses caractéristiques (montagne, colline, rivière, *etc*). Plusieurs méthodes permettent de créer des plantes, mais peu parmi elles modélisent des feuilles et surtout une variété de feuilles. Dans cette thèse, nous proposons des méthodes pour la génération et la modélisation contrôlée des objets naturels, en nous concentrant sur les caractéristiques d'un terrain ainsi que les feuilles afin de répondre à ces problématiques. Nous introduisons des techniques de génération combinée avec les techniques d'édition afin de bien contrôler nos résultats tout en préservant l'aspect naturel de l'objet généré. Nos contributions sont présentées dans trois chapitres.

Dans les chapitres 2 et 3, nous proposons deux modèles multirésolutions, le premier est basé sur les IFS (*Iterated Function System*) [Bar88], tandis que le deuxième est basé sur la technique de surface de subdivision [CC78]. La notion de détails comme celle employée dans la transformée en ondelette [Mey87, Mal89] est intégrée dans les deux modèles pour avoir plus de contrôle sur l'objet traité. Un formalisme général est introduit pour unifier les deux modèles sous une seule formule dont voici la définition :

$$\mathcal{P}^{J+1} = \varphi(\mathcal{P}^J) \oplus \psi(\mathcal{P}^J, \delta\mathcal{P}^J) \tag{1}$$

$\mathcal{P}^J$ représente l'objet dans un niveau de résolution J. $\delta\mathcal{P}^J$ représente le détail associé à ce niveau. $\varphi(\mathcal{P}^J)$ représente la fonction d'échelle que nous utilisons pour créer l'objet au niveau $J+1$ à partir de l'objet au niveau J. $\psi(\mathcal{P}^J, \delta\mathcal{P}^J)$ est la fonction de détail (fonction d'ondelettes dans le formalisme ondelettes), en plus de $\delta\mathcal{P}^J$, elle prend aussi en entrée l'objet $\mathcal{P}^J$. $\oplus$ est un opérateur utilisé pour combiner les résultats de $\varphi(\mathcal{P}^J)$ et $\psi(\mathcal{P}^J, \delta\mathcal{P}^J)$. Cette opération est appelée synthèse multirésolution (Figure 1). Ainsi, un objet $\mathcal{P}^N$ peut être représenté par sa version à la résolution J ainsi que les détails $\delta\mathcal{P}^J \dots \delta\mathcal{P}^{N-1}$ qu'il faut combiner successivement pour arriver à la résolution N. Les expressions précises de $\mathcal{P}^J$, $\delta\mathcal{P}^J$, $\varphi(\mathcal{P}^J)$, $\psi(\mathcal{P}^J, \delta\mathcal{P}^J)$ et $\oplus$ seront définies dans chacun des deux modèles. Dans

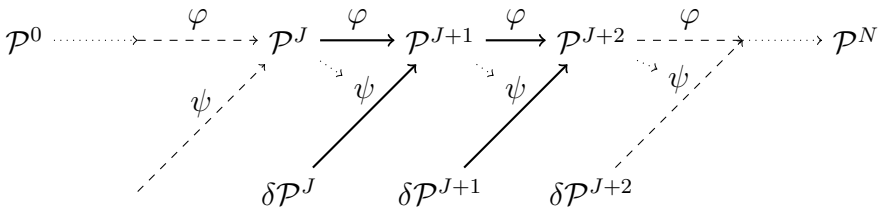


FIG. 1 – Synthèse multirésolution

les deux cas nous essayons de trouver une formule inversible, c'est-à-dire qui permette de faire l'opération dite d'analyse (Figure 2). L'analyse permet, à partir d'un objet à une résolution N, de le décomposer en une représentation multirésolution combinant les informations de détails $\delta\mathcal{P}^0 \dots \delta\mathcal{P}^{N-1}$ avec sa version basse résolution $\mathcal{P}^0$.

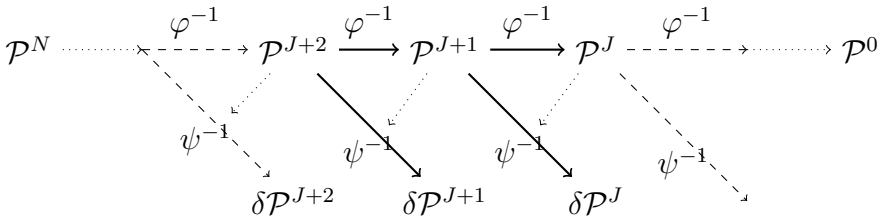


FIG. 2 – Analyse multirésolution

Dans le chapitre 2, notre modèle multirésolution combine les IFS (*Iterated Function System*) avec l'idée de détail comme celle utilisée dans la transformée en ondelette. L'avantage d'un tel modèle est la possibilité de combiner les nombreux avantages introduits par les modèles multirésolutions (édition , visualisation, *etc*), avec la capacité des modèles fractals de générer des formes rugueuses. Ce modèle a été utilisé pour représenter des objets rugueux en appliquant des déformations globales sur leur forme. Une étape d'optimisation est nécessaire pour que le modèle s'adapte à un objet donné. Ce modèle présente un outil simple et efficace pour l'analyse comme pour la synthèse des objets mais son étape d'optimisation le rend très coûteux en temps de calcul. De plus, il fournit un contrôle global ainsi qu'un contrôle local difficile à gérer ce qui ne nous convient que partiellement. Les inconvénients de ce modèle nous donnent la motivation pour introduire un autre modèle multirésolution (Chapitre 3) basé sur l'idée de subdivision tout en gardant la possibilité de générer des formes rugueuses.

Dans le chapitre 3, en prenant en compte tous les avantages et les inconvénients du modèle proposé au chapitre 2, nous proposons un modèle multirésolution basé sur la surface de subdivision. Comme l'autre modèle, ce modèle combine cette idée avec l'idée de détails. Nous modifions l'algorithme de subdivision pour être capable de produire des formes

fractales. Deux outils de contrôle sont présentés avec ce modèle : les masques utilisés pour subdiviser l'objet, et les points de contrôle que nous pouvons modifier interactivement à n'importe quel niveau de résolution. Le premier outil permet de modifier la forme fractale générée tandis que le deuxième permet de modifier la forme de l'objet localement, à des niveaux de résolution variables. Par exemple, durant la modélisation d'un terrain, la montagne créée dans un niveau grossier de résolution est plus étendue que celle créée dans un niveau plus raffiné.

En fin dans le chapitre 4, en se concentrant sur le problème de la génération de terrain, nous proposons une méthode permettant de contrôler localement la génération de terrain représenté par un champ de hauteur. L'idée principale de cette approche est la création de terrain en partant d'un ensemble de courbes représentants les caractéristiques (montagne, rivière, colline, falaise, *etc*). Ces courbes sont appelées courbes caractéristiques. Les courbes représentent des primitives génériques qui sont capables de modéliser une grande variété de caractéristiques d'un terrain. Ces courbes sont définies de manière vectorielle afin de pouvoir générer un terrain dans n'importe quelle résolutions. Un ensemble de points de contrainte est attaché à chaque courbe. Chacun de ces points possède un ensemble de paramètres (hauteur, angle de pente à gauche et à droite, rugosité, *etc*). Ces paramètres sont discrétisés sur deux images RGBA[1] de texture afin d'exploiter la capacité de parallélisme introduite par les GPUs[2] modernes. La première image contient les informations du gradient tandis que la deuxième contient les contraintes de hauteur et les paramètres de bruit. Deux passes de diffusion sont accomplies, la première diffuse l'image de gradient pour remplir les trous résultant de l'intersection des courbes. La deuxième passe calcule en même temps une carte des paramètres de bruit en utilisant la diffusion de Laplace ainsi qu'une carte d'élévation en le guidant par le gradient résultant de la première passe de diffusion. Une carte de détails est générée en utilisant la carte des paramètres diffusés de bruit. Le champ de hauteur final représentant le terrain est obtenu en ajoutant la carte d'élévation lisse et la carte de détails. Tous les algorithmes de diffusion sont calculés interactivement grâce à une implémentation parallélisée sur GPU[2].

Dans l'ensemble de nos modèles proposés, nous nous concentrons sur le problème de contrôle de la génération d' objets naturels et plus précisément le contrôle local. Nos modèles nous permettent de générer de terrain en contrôlant ses caractéristiques locales ainsi que la génération des variétés d'objets. Après avoir abordé les méthodes existantes, nos approches proposées serons détaillées dans la suite de ce document.

[1] *RGBA= Red Green Blue Alpha, les 4 composantes standards d'une image*
[2] *Graphic Processing Unit*

Chapitre 1

État de l'art



Sommaire

La génération de formes naturelles ainsi que le contrôle de ces formes sont des domaines vastes de recherche dans l'infographie car ces formes sont utilisées de manière intensive dans les paysages naturels. Dans ce chapitre nous abordons l'histoire de la génération et du contrôle de formes naturelles. Nous pouvons classifier les méthodes utilisées pour ce but en trois catégories : les méthodes procédurales, les techniques d'édition et les méthodes de simulation.

Les méthodes de génération procédurale permettent de créer rapidement et automatiquement une très grande quantité d'objets naturels variés à partir de très peu de paramètres. Même si ces méthodes sont très efficaces par rapport à la rapidité et à la diversité, le contrôle du résultat reste très difficile.

Le plupart des techniques d'édition emploie la méthode d'esquisses ou à base d'exemples réels afin de contrôler la forme des objets. Ces techniques exigent une forte intervention de l'utilisateur qui devient longue et fastidieuse lors de la génération d'un nombre important d'éléments. Malgré leurs inconvénients ces méthodes sont les plus utilisées dans un but de contrôle.

Finalement, les méthodes de simulation essayent de simuler et faire évoluer l'influence d'un phénomène sur un élément au cours du temps. L'idée principal de ces méthodes est la reproduction des phénomènes physiques appliqués à un objet afin d'obtenir un résultat réaliste. Ces méthodes emploient des équations physiques guidées par un certaine nombre de lois physiques ce qui donne un contrôle faible sur le résultat final. L'inconvénient majeur de ces méthodes est que la création d'un objet est longue et coûteuse car elle nécessite de résoudre de nombreuses équations physiques.

Ces trois catégories ne sont pas séparées, il y a toujours des méthodes qui se trouvent aux frontières afin d'en combiner les différents avantages. Ces méthodes cherchent à simplifier le modèle d'une catégorie et utiliser ce modèle simplifié dans une autre catégorie. Par exemple, une méthode qui emploie la génération procédurale et la simulation physique, utilise une version simplifiée des équations physiques de simulations afin d'accélérer les temps de génération procédurale.

Nous allons aborder dans ce chapitre les travaux qui sont faits dans chacune de ces catégorie en présentant leurs avantages et inconvénients. Nous ne considérerons pas les méthodes de simulation dans cette études parce qu'elles ne présentent que peu de contrôle sur le résultat final.

1.1 Génération procédurale

La génération procédurale est un terme largement utilisé dans la génération de formes naturelles, il se réfère aux objets générés algorithmiquement plutôt que manuellement. Souvent, cela signifie la création des objets à la volée c'est-à-dire sans stocker les objets. Cette technique consiste à générer des modèles géométriques en utilisant des algorithmes ou des règles. Le modèle obtenu n'a pas besoin d'être stocké car il peut être généré à chaque instant grâce à ces techniques.

1.1.1 Génération de terrains

Très souvent le terrain est l'élément dominant dans les scènes naturelles. C'est pourquoi la génération de terrain est un sujet de recherche depuis longtemps. Il y a plusieurs techniques permettant de reconstruire un terrain à partir de données réelles, mais les techniques les plus utilisées dans l'infographie sont les méthodes de génération procédurale qui permettent d'obtenir rapidement des terrains artificiels. Les modèles les plus utilisés pour représenter un terrain sont les surfaces 2.5D (champ de hauteur) en raison de leur structure simple, mais une surface 3D peut représenter beaucoup mieux les caractéristiques du terrain.

Génération sans contraintes

Nous pouvons constater que les méthodes de génération de terrain ont commencé avec les travaux de Mandelbrot et Van Ness [MVN68] dans lesquels le concept de fBm (fractional Brownian motion) est introduit. Mandelbort a remarqué la similarité entre la trace de fBm au fil du temps et la ligne de crête des montagnes [Man77]. En généralisant le processus en dimension 2, on obtient une surface Brownienne qui donne une approximation visuelle des montagnes dans la nature. Dorénavant les fBm sont utilisés largement afin de générer et rendre les terrains [FFC82b, Lew87a, Mil86]. La technique originale de génération de terrain qui est employée par Mandelbrot [Ben82] et Voss [Vos85] a été les *Poisson faulting*. Cette technique consiste à appliquer des déplacements aléatoires de Gauss à un plan ou à une sphère aux intervalles distribuées selon la distribution de Poisson. Le résultat est une surface brownienne. Cette approche a été utilisée pour créer des planètes fractales par Mandelbrot et Voss [Ben82].

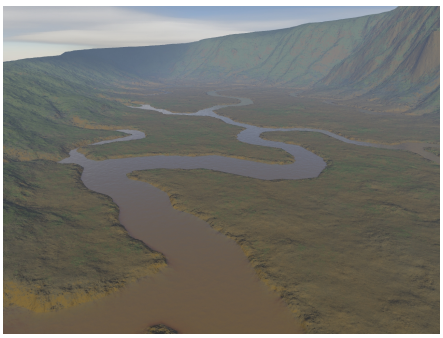
Dans [Sau88] une technique est proposée pour générer un fBm en prenant la transformée de Fourier d'un bruit blanc de Gauss en deux dimensions, puis en la multipliant dans l'espace des fréquences avec un filtre approprié, et finalement la transformée inverse de Fourier du produit est interprétée comme un champ de hauteur. Les avantages de cette approche incluent la disponibilité des lacunarités arbitraires et un contrôle précis du contenu fréquentiel global. Les inconvénients sont la périodicité de la surface finale, et l'absence de contrôle local de détail.

Les méthodes de déplacement du point milieu ont été introduites pour la communauté graphique comme une technique efficace de génération de terrain par Fournier, Fussell, et Carpenter [FFC82a]. Le schéma connu de subdivision de triangle [FFC82a] est premièrement introduit, il implique l'interpolation entre deux points dans le processus de subdivision. L'interpolation de trois points ou plus non-colinéaires est utilisé dans le schéma de *diamond-square* de Miller [Mil86], le schéma carré de Fournier et al [FFC82a], et la subdivision hexagone de Mandelbrot et Musgrave [Man88]. La subdivision stochastique généralisée [Lew87b] interpole plusieurs points locaux, contraints par une fonction d'autocorrélation. Miller [Mil86] a également proposé un schéma non imbriqué *square-square* de subdivision.

Depuis que Perlin l'a introduit il y a plus de vingt-cinq ans [Per85], le bruit a trouvé une large utilisation dans l'infographie. Plusieurs approches de génération de terrain fondées sur le bruit de Perlin sont introduites. Des terrains qui synthétisés par bruit ont été utilisés par Miller [Mil86], Gardner [Gar88] et Saupe [Jur89]. Miller a utilisé le bruit procédural de Perlin [G.S88] comme une carte de déplacements [Coo84] pour ajouter des détails sur les arêtes de polygones qui représentent les facettes d'une surface brownienne de contenu spectral similaire. Gardner a interprété sa fonction de bruit basée sur une série de Fourier [Gar85] comme un champ de hauteur. La quantification des valeurs d'altitude du champ de hauteur permet d'imiter des terrasses, telles que les mesas. Musgrave et al.[MKM89] utilise ce bruit combiné à un processus d'érosion hydraulique et thermique pour donner un aspect plus naturel au terrain généré.

Toutes les méthodes présentées ci-dessus ont l'avantage de générer de la variété de terrains en jouant avec les paramètres d'entrée. L'inconvénient commun de ces méthodes est l'incapacité de contrôler leur résultat final, de plus, quelques méthodes produisent des formes différentes même avec les mêmes paramètres d'entrée.

Génération avec contraintes



La génération procédurale de terrain est une méthode très efficace pour créer rapidement un terrain avec peu d'information. Pour cette raison, de nombreuses recherches sont effectuées pour contrôler ce type de génération. Prusinkiewicz et al. [PP93] a introduit une méthode pour générer des montagnes avec des rivières. Leur méthode intègre un modèle de courbe du cours d'une rivière dans le modèle de déplacement du point milieu. La méthode est basée sur l'observation que les deux modèles peuvent être exprimés par des mécanismes contextuels similaires de la réécriture. Belhadj [Bel07] a présenté un algorithme contrôlable basé sur les fractales pour reconstruire des cartes d'élévation. Cet algorithme peut créer des cartes d'élévation conformément à des contraintes. Aussi, avec cette méthode, l'utilisateur final peut donner ou modifier les principales caractéristiques, détails locaux et la morphologie de sa carte d'élévation et obtenir instantanément la surface de terrain. Dans [BA05b, BA05a] une extension de la méthode de déplacement du point milieu est introduite afin d'intégrer des réseaux d'eau dans la génération de terrain. Stachniak [SS05] a présenté une approche qui emploie une recherche stochastique afin d'identifier une série de modifications locales. Ces modifications déforment le terrain fractal afin de se conformer à un ensemble de contraintes. Les résultats présentés montrent que la méthode peut incorporer de multiples contraintes simultanément, tout en préservant l'aspect naturel du terrain fractal.

Représentation volumique

Il est impossible de modéliser certaines caractéristiques largement présentes dans la nature telles que des abris sous roches, des surplombs ou des cavernes avec la technique des champs de hauteur dite 2.5D, pour représenter le terrain. Avec ce type de représentation, le maillage du terrain est obtenu en utilisant une discrétisation régulière sous forme de grille et les sommets sont déplacés verticalement. Ce problème a été partiellement résolu par Gamito et Musgrave [GM01] pour la création de surplombs. Leur méthode consiste à utiliser un champ vectoriel permettant d'appliquer des déformations non linéaires à la surface d'un terrain obtenu par élévation. Cependant, il est

très difficile de contrôler avec exactitude la déformation et par conséquent la forme du surplomb.

Peytavie et al. [PGMG09] ont proposé un modèle hybride combinant une représentation discrète et une représentation implicite permettant de synthétiser des terrains complexes de topologie et de géométrie quelconques. Leur approche permet de représenter des surplombs, des grottes, des cavernes, des arches et définir les différents matériaux du sol tel que les pierres, la terre ou le sable. Les scènes sont éditées interactivement par de la sculpture, du dépôt de matière ou par des outils de simulation permettant de modifier la scène avec des outils de haut niveau. Les matériaux granuleux tels que le sable et les cailloux sont stabilisés à l'aide d'un algorithme d'angle au repos. La surface du terrain est extraite par produit de convolution et est texturée par projection et mélange de textures. Cependant, La création des terrains de plusieurs dizaines de kilomètres est très difficile car l'approche édition peut la rendre laborieuse. De plus, la représentation volumique a un coût non négligeable en mémoire.

Conclusion

Les modèles fractals sont les modèles les plus utilisés pour la création de terrain grâce à leur représentation compacte. Avec ces modèles, la création des terrains est accomplie en utilisant quelques paramètres ainsi que les algorithmes associés. Pour cette raison, plusieurs logiciels commerciaux utilisent ce type de modèles. Les modèles fractals avec des contraintes présentent aussi un moyen pour contrôler la création de terrain en préservant la simplicité de la représentation de terrain. Même si ces modèles peuvent générer facilement des terrains, leur représentation 2.5D ne leur permet pas de présenter plusieurs caractéristiques plus complexes qui nécessitent une représentation 3D. Malgré le fait que la représentation volumique permet de modéliser des caractéristiques complexes de terrain en présentant plus de contrôles sur la forme de terrain, la structure de données de cette représentation doit être stockée et ne peut pas être générée par des algorithmes procéduraux.

1.1.2 Génération de végétation

Les plantes ainsi que les arbres sont des éléments importants à prendre en compte dans les paysages naturels. La structure complexe de tels éléments les rend difficiles à modéliser de manière réaliste. Par contre la similarité dans la structure de ces éléments encourage l'utilisation des modèles autosimilaire afin de les modéliser. Les modèles le plus connus parmi ces modèles sont les L-Systèmes [Lin68] et les IFS (Iterated Function System) [Bar88, Bar93].

L-Système

Un L-Système est une grammaire formelle, permettant un procédé algorithmique, inventé en 1968 par le biologiste hongrois Aristid Lindenmayer [Lin68] qui consiste à modéliser le processus de développement et de prolifération de plantes ou de bactéries. À l'origine, les L-Systèmes ont été inventés pour fournir une description formelle du développement des organismes multicellulaires simples, et pour illustrer les relations de voisinage entre les cellules végétales. Mais plus tard, avec les progrès dans l'informatique, ces systèmes ont été étendus pour décrire les structures des plantes supérieures et des branchements complexes [PL91]. Un L-Système est un ensemble de règles et de symboles qui modélisent un processus de croissance d'êtres vivants comme des plantes ou des cellules. Le concept central des L-Systèmes est la notion de réécriture. La réécriture est une technique pour construire des objets complexes en remplaçant des parties d'un objet initial simple en utilisant des règles. Étant donnée une chaîne de symboles initiale, celle-ci est itérativement réécrite en utilisant les règles de la grammaire pour obtenir une chaîne plus complexe. Cependant, la structure générée est beaucoup trop régulière pour obtenir des plantes réalistes. Ce problème a été résolu en utilisant des L-Systèmes paramétriques introduits par Prusinkiewicz et Lindenmayer [PL91]. Le principe consiste à ajouter des paramètres de contrôle aux règles ainsi qu'une dépendance au prédécesseur. Dorénavant, de nombreux travaux sont introduits pour augmenter la réalisme des plantes créées par ces systèmes et offrir plus de contrôle à l'utilisateur [BPF+03]. [PHM93] une méthode est proposé pour l'animation du développement des plantes en intégrant les L-systèmes et les équations différentielles. Des méthodes [PJM94, MP96, RLP07, PHL+09] pour générer les plantes

en prenant en compte leurs interactions avec les environnements sont introduites afin de modéliser les écosystèmes. Bien que les L-systèmes soient des modèles très efficaces pour créer des objets complexes, leur grand inconvénient est que le choix des règles de production n'est pas intuitif et dans la plupart des cas il y a besoin d'un expert pour le faire.

IFS (Iterated Function System)

En mathématique, les IFS sont des méthodes de construction de fractales, les résultats de cette construction sont toujours autosimilaires. Les IFS peuvent être de n'importe quelle de dimension. L'objet fractal est constitué de l'union de plusieurs copies de lui-même, chaque copie étant transformée par une fonction. Ce modèle a été étudié par Hutchinson [Hut81] en appliquant le théorème du point fixe aux ensemble de compacts d'un espace métrique complet. Barnsley [Bar88] a développé ce modèle et l'a utilisé pour la génération fractale d'objets naturels. Plusieurs travaux fondés sur les IFS sont introduits afin de créer des plantes. Barnsley [Bar88, Bar93] a employé un IFS consistant en quatre fonctions contractantes afin de générer une fougère. Han [Han07] a utilisé une méthode basée sur la fusion entre L-système et IFS dans laquelle le L-système est utilisé afin de simuler la structure topologique de la plante et les IFS servent à simuler les composantes de la plante.

L'IFS projeté est une variante des IFS introduite par Zaïr [ZT96]. Ce modèle utilise le principe employé par les formes à pôles comme Bézier par exemple. L'idée principale des IFS projetés est la séparation entre l'espace d'itération et l'espace de modélisation. Une forme à pôles consiste en deux parties : les points de contrôle et les fonctions de mélange. Les fonctions de mélange dans les IFS projetés deviennent des modèles IFS. Zaïr [ZT96] a utilisé ce modèle pour la modélisation d'objets fractals. Guérin [GTB02] a employé ce modèle afin de modéliser et approximer des courbes et surfaces fractales.

La représentation mathématique de ces modèles permet de générer des objets naturels complexes en utilisant des fonctions mathématiques simples. Mais le choix des paramètres de fonctions n'est pas évident et nécessite un expert pour bien les choisir, ainsi que l'approximation d'un élément réel nous exige de faire face au problème inverse qui est un problème ouvert et difficile à résoudre.

Modèles non-fractals

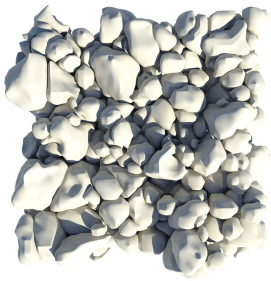
Les modèles autosimilaires ne sont pas les seuls modèles existants pour créer des plantes. Il y a d'autre méthodes qui ne dépendent pas de tels modèles et emploient d'autres approches. Weber et al. [WP95] ont introduit un modèle qui crée et rend un arbre en utilisant des paramètres géométriques. Les branches de l'arbre sont définies par des objets géométriques comme des cônes par exemple. Les niveaux de récursivité de l'arbre sont limités à trois ou quatre niveaux, pour chacun de ces niveaux un ensemble de paramètres est défini afin de créer des arbres complexes géométriquement. Une autre approche géométrique est proposée par Deussen et al. [DL97]. Leur méthode consiste à encapsuler des informations structurales et géométriques dans des objets. Ces objets sont combinés selon de règles pour former une description du modèle de plante. La plante sera ensuite générée facilement et rapidement en utilisant ce modèle. Bien que les résultats de cette approche sont complexes et réalistes, il est difficile d'avoir le résultat souhaité à cause du grand nombre de paramètres utilisés.

Une méthode qui utilise des propriétés botaniques de plantes est introduite par De Reffye et al. [dREF+88]. Le modèle proposé intègre les connaissances botaniques de l'architecture des arbres (comment ils se croissent, occupent l'espace, où et comment les feuilles, les fleurs ou les fruits se trouvent, *etc*). L'intérêt de ce modèle est sa grande richesse : les mêmes méthodes procédurales peuvent produire une grande variété des plantes (sapins, cèdres, frangipaniers, peupliers, pins, merisiers, herbes, *etc*). Un autre avantage très important qui peut être tiré de ce modèle est l'intégration du temps qui permet de visualiser le vieillissement d'un arbre. Streit et al. [SFS05] ont introduit une méthode basée sur la biologie qui imite les fondements de la variation des plantes réelles en utilisant un modèle donné (L-systèmes) des végétaux. Cette méthode utilise un système de contrôle de rétroaction afin de simuler le mécanisme de la croissance biologique par lequel une plante réagit naturellement à des facteurs environnementaux. Cette technique crée des modèles variés de façon plus réaliste par la modélisation des réponses de croissance à des stimulations, et fournit une méthode pour créer rapidement de nombreux modèles similaires, aucune d'entre elles ne sont exactement semblables.

Conclusion

La génération de plantes réalistes a été le sujet de nombreuses recherches durant des dizaines d'années. Les L-systèmes ainsi que les IFS sont utilisés avec un grand succès pour créer des plantes et des arbres. D'autres modèles basés sur les caractéristiques botaniques sont aussi utilisés avec succès pour générer les plantes. Le grand inconvénient de ces modèles est que le choix de leurs paramètres est une tâche très difficile et a besoin des experts pour l'accomplir. De plus, certains modèles génèrent un grand nombre de polygones qui doivent être stockés en mémoire ce qui rend la création de scènes complexes contenant des plantes et des arbres très coûteuse.

1.1.3 Génération de cailloux



Les roches et les pierres présentent une grande variété d'apparences dans les paysages. Généralement ces éléments sont considérés comme des parties du terrain généré avec les méthodes de génération de terrain sans prendre en compte leur propre structure. La présence de cailloux et de pierres de manière individuelle devient une exigence dans un but de réalisme pour un terrain de petite échelle. En effet il n'existe pas beaucoup de méthodes qui génèrent des cailloux procéduralement. Ces éléments sont largement modélisés individuellement ce qui est long et coûteux pour la génération de paysage naturel. Des méthodes basées sur la technique de déplacement du point milieu sont introduites dans [Dei03, Dac06] afin de générer des cailloux. La génération procédurale commence par un nombre relativement faible de points aléatoires, par exemple, 4 à 20 points qui définissent la forme approximative de la roche. Ensuite ces méthodes calculent l'enveloppe convexe de ces points pour obtenir une triangulation initiale. Finalement, la triangulation initiale est subdivisée une à quatre fois et les sommets nouvellement insérés sont perturbés avec des fonctions de bruit appropriées. Les étapes de subdivision augmentent les détails de la surface et l'apparence naturelle de la roche.

Peytavie et al. [PGGM09] sont les premiers qui introduisent une méthode complètement procédurale pour générer une variété de cailloux. Leur approche repose sur un algorithme de coin de cube modifié pour produire un ensemble de tuiles apériodiques. La méthode de construction est généralisée de telle sorte que la géométrie des roches doit chevaucher les coins de cubes en vue d'éviter les lacunes irréalistes dans l'agencement des roches. En outre, une technique originale est proposée pour contrôler la forme des roches

en contact par le calcul des cellules de Voronoï en utilisant une distance paramétrée anisotrope. Cette méthode a été utilisée pour générer des paysages et des huttes de pierre ainsi des que murs avec des milliers de rochers entassées.

Conclusion

Quelques méthodes sont proposées pour la génération procédurale de cailloux. Les méthodes qui sont basées sur l'algorithme de déplacement du point milieu peuvent générer des cailloux procéduralement mais le nombre de polygones augmente très vite et les polygones risquent de s'intersecter à cause du déplacement aléatoire des nouveaux points. Bien que la méthode de tuiles apériodiques soit une méthode très efficace pour générer des milliers de roches à la fois, les résultats de génération sont des cailloux qui ont quasiment tous la même taille.

1.2 Les méthodes d'édition

Les méthodes d'édition sont très efficaces concernant le contrôle des formes des objets parce qu'elles permettent l'interaction de l'utilisateur avec les objets modélisés. Il y a une grande variété de ces méthodes (déformation, création, édition interactive, *etc*) mais nous nous concentrons sur les méthode d'édition pour la génération. Nous pouvons classer les techniques d'édition pour la génération en trois catégories : les techniques d'esquisses, la génération à partir de données réelles et la génération à partir d'images.

La génération par esquisses consiste à tracer en deux dimensions la forme souhaitée d'un objet. Puis, un algorithme traite ces esquisses pour générer le maillage en trois dimensions. Le maillage n'est pas forcément généré totalement, parfois une partie du maillage est généré localement.

Parfois, les données d'un exemple réel sont connues mais elles sont incomplètes ou bien clairsemées. De nombreuse méthodes sont introduites afin de générer un objet ressemblant à l'exemple original à partir de telles données.

Les techniques de génération à partir d'images reconstruisent un objet en utilisant une ou plusieurs images prises de cet objet de différents points de vue. Ces techniques produisent un objet très proche de l'image exemple de cet objet et minimisent l'intervention de l'utilisateur durant la génération.

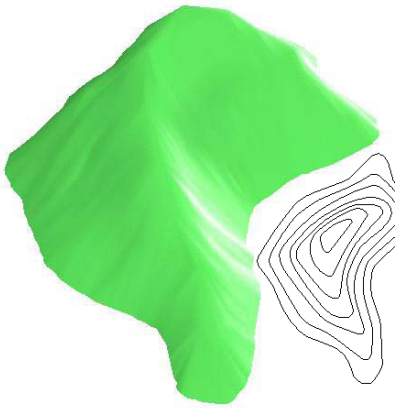
1.2.1 Terrain

Les méthodes d'édition sont utilisées dans la génération de terrain afin de contrôler le résultat final de la génération.

Esquisses

Il n'y a pas beaucoup de méthodes pour la génération de terrain par esquisse. Une méthode proposée par Cohen et al. [CHZ00] utilise cette technique afin de créer un monde 3D. L'utilisateur peut directement tracer la ligne de crête pour créer des collines et des montagnes, mais le terrain résultant semble mal conçu depuis d'autres points de vue. Une interface d'esquisses est proposée par Watanabe et al. [WI04] et peut générer un modèle de terrain dont la ligne de crête correspond aux esquisses tracées à l'écran. Whelan et al. [WV03] ont introduit une méthode qui modélise un terrain en utilisant les silhouettes des objets désirés définies par un ensemble d'esquisses. Une méthode efficace pour la génération de terrain par esquisse est proposée par Gain et al. [GMS09]. Cette méthode permet aux utilisateurs de dessiner la silhouette, la crête et les courbes englobantes des deux types de reliefs : les extrudés (collines et montagnes) et les intégrés (lits de rivière et canyons). Le terrain est généré de manière interactive pour répondre aux contraintes posées en utilisant la déformation de surface multirésolution. En outre, les caractéristiques du bruit d'ondelette [CD05] de la silhouette se propagent au terrain environnant.

À partir de données réelles

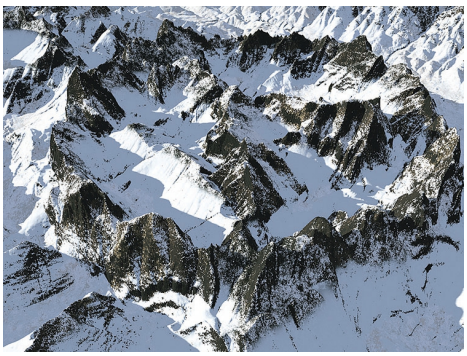


La reconstruction de terrain en utilisant de données clairsemées mais réelles, comme les cartes de topographie, est un sujet de recherche attaqué par nombreux chercheurs. Chai et al. [CMN98] ont introduit une méthode qui reconstruit une surface lisse avec la continuité C^1 partout sur le terrain en utilisant des gradients guidés par des équations aux dérivées partielles. Une méthode qui est basée sur l'extraction de squelette ou l'axe médian est présentée par Thibault et al. [TG00]. Des points additionnels dont la hauteur estimée, de ce squelette sont utilisés pour éliminer les cas des triangles ayant des angles très aigus. Cette méthode garantit de préserver les relations topologiques entre les segments de la courbe

dans la triangulation irrégulière finale. Une interpolation hermitienne est employée dans [HSS03] pour calculer la hauteur des points d'axe médian utilisés dans la triangulation finale. Les surfaces générées sont de continuité C^1 partout, sauf aux caractéristiques de terrain comme les crêtes et les vallées. Byeong-Seok et al. [SJ04] ont proposé une méthode de reconstruction rapide, qui génère des bandes de triangles avec une seule opération pour la région simple qui ne contient pas de branches. S'il y a des branches dans les contours, les courbes de niveau sont partitionnées en plusieurs sous-contours. Des méthodes similaires [SJ05, RdlF05] sont proposée afin de reconstruire le terrain à partir des contours. La triangulation finale est aussi une triangulation irrégulière. Bien que les résultats de ces méthodes soient convaincantes d'un point de vue réaliste, l'utilisation d'une grille régulière pour représenter le terrain reste meilleur que la triangulation irrégulière utilisée avec les méthodes ci-dessus.

Pouderoux et al. [PGTG04] présente un algorithme rapide pour regénérer le terrain par l'interpolation et l'approximation lisse des données d'élévation dispersées. La surface globale est reconstruite en la subdivisant en sous-domaines qui se chevauchent localement en utilisant un arbre binaire équilibré. Dans chaque feuille de cet arbre, une surface lisse locale est reconstruite en utilisant des fonctions de base radiale. Enfin un mélange hiérarchique est fait pour créer la surface finale de continuité C^1. L'avantage de cette méthode est l'utilisation d'une grille régulière pour représenter le terrain. Une méthode qui emploie aussi une grille régulière est récemment proposée par Zhuo et al. [CZ09]. Cette méthode exploite le GPU pour reconstruire le terrain à partir de contours en temps réel.

À partir d'un exemple



Certaines méthodes sont proposées pour générer le terrain à partir d'un exemple ou bien une image des caractéristiques de terrain. Chiang et al. [MJW+05] présentent une technique qui modélise le terrain en utilisant des primitives géométriques simples de terrain (triangle, trapèze, *etc*), puis des caractéristiques qui conviennent le mieux à ces primitives sont obtenues à partir d'une base de données réelles. Une méthode similaire à celle-ci est introduite par Shih-Chun et al. [THT08], les primitives géométriques utilisée sont plus adaptées aux caractéristiques de terrain. Ces primitives sont remplacées par des unités de terrain de données réelles d'élévation. Brosz et al. [BSS06] ont introduit une méthode pour synthétiser le terrain à partir d'un exemple réel de terrain représentant des carac-

téristiques. La méthode commence par construire une forme de base globale du terrain désiré en résolution basse, puis une analyse multirésolution est appliquée sur le terrain réel pour extraire les détails de haute résolution. Enfin la méthode augmente la résolution du terrain de base en ajoutant ces détails. Zhou et al. [ZSTR07] se sont inspirés des méthodes de synthèse de textures à partir d'un exemple et plus particulièrement de celle proposée par Wu et al. [WY04]. Les terrains exemples (représentés par un champ de hauteur) sont utilisés pour générer de nouveaux terrains. La synthèse est guidée par des esquisses de l'utilisateur qui spécifie où les caractéristiques du terrain se produisent sur le terrain final. Les terrains exemples et la carte d'esquisses de l'utilisateur sont stockés sous forme d'images. Ces images sont découpées en morceaux rectangulaires puis le système cherche les morceaux des images exemples qui correspondent aux caractéristiques qui se retrouvent dans la carte d'esquisses. Enfin les morceaux sont assemblés à l'aide d'un algorithme minimisant le coût des différences d'altitude le long du chemin de jointures. Ces jointures sont finalement lissées en utilisant les équations de Poisson pour créer des transitions sans discontinuité.

Conclusion

Les différentes méthodes d'édition présentent un grand contrôle sur la forme finale du terrain. Les méthodes d'esquisses fournissent à l'utilisateur un moyen très efficace pour modéliser interactivement la forme de terrain en ajoutant plusieurs caractéristiques. Ces caractéristiques sont modélisées en les traçant sur l'écran. Le grand inconvénient de ces modèles est qu'ils ne sont pas réutilisables c'est-à-dire qu'une fois que le terrain est créé, il ne peut pas être utilisé pour créer d'autres terrains. Bien que les méthodes qui sont basées sur des données réelles sont capables de reconstruire des terrains très proches des données d'entrée, elles servent à reconstruire les terrain sans introduire de moyen pour ajouter des autres caractéristiques. Finalement les méthodes basées sur les exemples ou sur les images peuvent générer des terrains qui présentent des caractéristiques fournies par des exemples réels mais ces méthodes sont généralement coûteuses en temps de calcul.

1.2.2 Plantes

Plusieurs méthodes sont proposées afin de contrôler la forme de plantes ou d'arbres. Des méthodes d'esquisse ou à partir d'une image sont utilisées par les chercheurs pour contrôler ou bien reconstruire la forme des plantes.

Esquisses

La génération basée sur des esquisses de plantes n'apparaît que dans les dix dernière années. Une des premières méthodes qui traitent ce domaine a été présentée par Okabe et al. [OOI05] dans laquelle le modèle 3D d'un arbre est généré à partir d'une esquisse de cet arbre en 2D tracée par l'utilisateur. Des branches et des feuilles additionnelles peuvent être ajoutées ou éditées puis la méthode les réplique sur tout l'arbre avec un mécanisme de propagation sans aucune information botanique ce qui rend parfois les résultats non-réalistes. Wither et al. [WBCG09] ont présenté une méthode permettant de dessiner la silhouette des feuillages à différents niveaux ce qui permet de contrôler la forme globale de l'arbre. Les connaissances botaniques sont intégrées dans la méthode, ce qui permet de déduire les branches qui relient à leur parent d'une manière plausible et d'avoir une distribution correcte en 3D. L'avantage de dessiner la silhouette des feuillages à différents niveaux dans cette approche, devient un moyen très fastidieux pour générer un nombre important d'arbres.

Ijiri et al. [IOOI05] ont introduit une interface pour modéliser des fleurs en 3D tout en préservant une structure botanique correcte des fleurs. L'utilisateur peut dessiner le diagramme de fleur qui représente les composantes d'une fleur, et le modèle de floraison qui arrange les fleurs ensemble. Le système utilise ces informations afin de créer le modèle final. L'intervention de l'utilisateur définit globalement la structure de fleur, pour cette raison Ijiri et al. [IOI06a] ont généralisé la création d'une plante en proposant d'employer une hiérarchie de plans. Chaque plan permet d'effectuer l'esquisse de chacune des parties de la plante avec la possibilité de sauvegarder l'esquisse pour une utilisation ultérieure. Ces deux méthodes donnent des résultats visiblement réalistes, mais la structure de plante dépend de l'expérience de l'utilisateur qui doit nécessairement interagir un important nombre de fois, et ne peut pas être extraite à partir d'une plante originale. De plus l'arrangement de plante est correct mais n'est pas guidé par des critères botaniques.

Streit et al. [SLSS06] ont proposé une interface permettant la modélisation de variété de plantes de manière procédurale par la spécification des branches et des tiges. Cette interface permet de dessiner d'esquisses 3D. Ces esquisses 3D sont ensuite utilisées avec un modèle de base de plante choisi par l'utilisateur par une méthode de simulation biologique permettant de créer une variété de plantes. Les modèles de base sont des modèles L-systèmes et ne peuvent pas être des modèles réels de plantes.

Ijiri et al. [IOI06b] ont proposé les premiers une méthode qui tente de contrôler la croissance d'un modèle L-système. Elle permet à l'utilisateur de contrôler l'apparence

globale et la profondeur de la récursivité de L-système. un nouveau module en L-système est introduit dont la direction de la croissance est guidé par l'esquisse de l'utilisateur. La simulation de la croissance suit progressivement l'esquisse tracée par l'utilisateur et crée un arbre le long de l'esquisse. Anastacio et al. [APS09] ont développé une méthode similaire permettant la spécification et la création des structures de plante en utilisant un modèle L-systèmes dont les caractéristiques globales et locales sont manipulées par une esquisse de l'utilisateur. La structure générale, la posture et les proportions de la plante sont d'abord esquissés par l'utilisateur. Ils sont automatiquement encodés comme un ensemble de fonctions de position qui contrôlent les longueurs entre nœuds, les angles de ramification, la taille des organes, et la tige. Ces fonctions de position sont ensuite utilisées pour paramétrer des modèles L-système prédéfinis représentant les motifs phyllotaxiques pour le positionnement des surfaces latérales des organes tels que les feuilles et les pétales.

Une approche probabiliste est proposée par Chen et al. [CNX+08]. Cette approche convertit une esquisse très grossière à un modèle 3D complet d'arbre en extrayant les paramètres initiaux de l'arbre d'une base de données en analysant la similarité entre l'esquisse et les modèles stockés. L'interaction des branches est modélisée par un champ de Markov, sous la contrainte de la projection 3D d'esquisse. Le système ensuite utilise l'auto-similarité pour ajouter de nouvelles branches avant de finalement peupler toutes les branches avec des feuilles choisies par l'utilisateur. Bien que cette approche permette de créer des variétés de modèles d'arbres avec un aspect naturel, elle oblige à créer une grande base de données afin de répondre à n'importe quelle esquisse fournie par l'utilisateur.

À partir d'un exemple

Plusieurs méthodes ont été proposées afin de créer des arbres à partir d'une ou plusieurs images d'arbres réels. Par exemple, la méthode introduite par Shlyakhter et el. [SRDT01] dirige la croissance d'un L-système, par un ensemble donné de photographies. Une enveloppe visuelle est reconstruite à partir de l'ensemble des images d'entrée enregistrées. Le diagramme de l'axe médian de l'enveloppe visuelle est construite et utilisée comme le squelette de l'arbre. Les petites branches et les feuilles sont ajoutées à l'aide d'un L-système. Les résultats obtenus sont très proches des images fournies en entrée mais possèdent un important nombre de polygones.

Bien qu'elle soient complexe, une approche très précise de modélisation basée sur l'image est décrite par Reche et al. [MMD04]. Dans ce cas, un ensemble de photographies

soigneusement enregistrées est utilisé pour déterminer la forme volumétrique d'un arbre donné. Le volume est divisé en cellules, pour chaque cellule, une représentation visuellement valable est calculée par un ensemble de textures. La série complète des textures représente assez fidèlement l'arbre. Bien que les résultats soient impressionnants, leur approche ne récupère pas la géométrie explicite des branches et des feuilles. En conséquence, leur technique est limitée à la visualisation uniquement, sans aucun moyen direct pour la modification. Toutefois, une grande quantité d'espace de texture de l'ordre de plusieurs dizaines de mégaoctets est nécessaire. Aussi, il n'est pas facile de montrer l'arbre sous différentes conditions d'éclairage car l'éclairage est déjà intégré dans les textures.

Une autre méthode de génération de plantes à base d'images pour la modélisation des petites plantes a été présentée par Quan et al. [QTZ+06]. Ici, une séquence d'images est enregistrée et le modèle est calculé à partir de ces images d'une manière semi-automatique. Bien que les résultats soient de meilleure qualité, la quantité d'entrées requise par l'utilisateur empêche la modélisation des objets complexes pour lesquels une telle méthode serait particulièrement intéressante.

Une approche pour générer des modèles 3D d'arbres d'aspect naturel à partir d'images a été proposée par Tan et al. [TZW+07]. Cette approche a un avantage supplémentaire car il ne nécessite qu'une petite intervention de l'utilisateur. Bien que cette approche soit essentiellement basée sur l'image, elle ne modélise pas chaque feuille directement à partir d'images à cause du grand nombre de feuilles. Au lieu de cela, l'arbre est peuplé avec des répliques de feuilles à partir d'images segmentées afin de reconstituer la forme globale de l'arbre. En outre, les formes de branches visibles sont utilisées pour prédire celles des branches obscurcies. Les résultats sont convaincants mais la similarité des feuilles et des branches reste un inconvénient de cette approche.

Neubert et al. [NFD07] ont introduit une méthode pour produire des modèles 3D d'arbre à partir de photographies. Un volume de voxel contenant la densité de feuillage est estimé à partir des informations des images d'entrée. Les valeurs de densité sont utilisés pour produire des positions initiales pour un ensemble de particules. En exécutant une simulation de l'écoulement 3D, les particules sont tracées vers le bas et sont combinées pour former des rameaux et des branches. La géométrie du squelette d'arbre est produite en utilisant des règles botaniques pour les épaisseurs de branche et les angles d'embranchement. Enfin, les feuilles sont ajoutées. Des initialisations différentes pour la simulation de particules conduisent à une variété dont les structures de branchement ressemblent à une seule série de photos.

Conclusion

De nombreuses méthodes ont été proposées afin de modéliser des plantes en utilisant l'esquisse. Ces méthodes modifient les règles de L-systèmes ou les caractéristiques botaniques utilisées pour créer des plantes selon les esquisses tracées par l'utilisateur. Un grand inconvénient de ces méthodes est que les esquisses ne sont pas génériques. Parfois, chaque famille de plantes a besoin de son propre algorithme. Bien que les méthodes basées sur un exemple réel sont capables de générer des résultats réalistes et très proches de l'exemple d'entrée, elles génèrent un grand nombre de polygones ce qui rend le modèle coûteux en mémoire, ou bien utilisent un motif qui est propagé en utilisant la similarité ce qui conduit à un modèle très similaire.

1.2.3 Feuilles

Bien que de nombreuses recherches aient été introduites pour la génération des plantes et d'arbres, quelques méthodes sont proposées pour traiter indépendamment la génération des feuilles des plantes. Prusinkiewicz et al. [PMKL01] ont fourni une représentation détaillée de l'interaction et la combinaison avec des algorithmes paramétrés pour la modélisation des plantes et la création des scènes réalistes impliquant des plantes, y compris les feuilles enroulées. Dans cette méthode un algorithme pour créer les feuilles est proposé dans un but de création de scènes naturelles de plantes.

Le travail proposé par Rodkaew et al. [RLFS02] présente une méthode qui combine deux techniques : L-systèmes et algorithmes génétiques (AG) pour rechercher une expression décrivant la réécriture des formes de feuilles. Un L-système est utilisé pour construire la forme d'une expression donnée de réécriture ainsi que l'AG pour rechercher des paramètres d'ajustement de l'expression de réécriture. Le remplacement des valeurs réelles de paramètres par des fonctions de balise est introduit. Le résultat montre que la méthode proposée produit une sortie acceptable mais un nombre important d'itérations est nécessaire pour que le L-système se rapproche d'une forme donnée de feuille.

Mündermann et al. [MMPP03] ont proposé une méthode de modélisation de feuilles lobées. Cette méthode étend le concept de balayages de squelettes branchés. L'entrée du modèle est une silhouette de feuille en 2D, qui peut être définie de façon interactive ou dérivée d'une image numérisée d'une feuille. L'algorithme calcule le squelette (axe médian)

de la feuille et l'approxime en utilisant des courbes splines interconnectées à une structure de ramification (splines collantes). La surface de la feuille est ensuite construite en balayant une courbe de production le long de ces splines. L'orientation de la courbe de production est ajustée afin de capturer correctement la forme du lobe de la feuille près des extrémités et des points de branchement du squelette, et d'éviter les auto-intersections de la surface.

Certaines méthodes sont proposées pour générer le système de nervures d'une feuille. Rodkaew et al. [RCSL04] présentent un système de transport de particules pour la modélisation de feuilles. Un ensemble de particules est initialisé aléatoirement dans une forme donnée d'une feuille. Chaque particule contient de l'énergie. La règle de transport dirige chaque particule vers une cible. Lorsque les particules sont à proximité, elles sont combinées. Les chemins de particules déplacées sont utilisés pour générer les modèles de nervures. Cette méthode est aussi adaptée à la production de couleurs dans les modèles des feuilles. La diffusion de la couleur se fait par le mouvement des particules. Le mélange de couleurs vives génère des motifs marbrés.

Hong et al. [HSB05] ont proposé une méthode interactive pour la modélisation de la surface de la feuille. Leur méthode débute en extrayant le contour d'une image de feuille. Puis le squelette est calculé à partir de ce contour. Le modèle d'une feuille composée de ce squelette en trois dimensions formé par les veines plus nombreuses et une surface représentant le lobe de la feuille. Les veines peuvent être modifiées de façon interactive pour créer la forme 3-D du modèle de la feuille. La représentation du modèle se compose de deux structures de données couplées, une structure de données arborescente des veines pour le squelette des feuilles et un maillage non structuré triangulaire pour la membrane de la feuille. Le squelette est modifié par l'utilisateur et le maillage de la membrane est un maillage de la surface qui suit la forme du squelette calculé par interpolation harmonique. L'inconvénient de cette méthode est qu'elle nécessite des interactions manuelles excessive pour générer une forme désirée de feuille.

Runions et al. [RFL+05] ont introduit une classe d'algorithmes basés sur la biologie pour générer des modèles de nervures. Ces algorithmes simulent l'interaction entre les trois processus : (1) le développement des veines vers les sources d'hormones incorporées dans le lobe de la feuille, (2) modification de la distribution de la source d'hormones par la proximité des veines, et (3) la modification de deux modèles : la veine et la source de distribution par la croissance des feuilles. Ces processus sont formulés en termes d'opérations itératives géométriques sur des ensembles de points qui représentent les nœuds de

veines et les sources hormonales. En outre, un graphe de connexion de veine est utilisé afin de déterminer les largeurs de veine.

Une méthode basée sur le squelette de nervure pour la modélisation et l'animation de flétrissement des feuilles est présenté par Shenglian et al. [SCX09]. La méthode proposée se compose de cinq principaux processus. Tout d'abord, un squelette en trois dimensions d'une feuille est construit à partir d'une image de la feuille, et le squelette des feuilles est également utilisé pour générer un maillage détaillé pour la surface de la feuille. Ensuite, un squelette de nervures est généré de manière interactive à partir du squelette de la feuille. Chaque veine dans le squelette de nervures se compose d'une chaîne de sommets segmentés. Troisièmement, chaque sommet du maillage de la feuille est connecté au plus proche sommet dans le squelette de nervation. Ensuite le squelette de nervures est déformé en contrôlant le mouvement de chaque sommet dans le squelette de nervures en le faisant tourner autour d'un vecteur fixe. Enfin, le maillage de la feuille est mis en correspondance avec le squelette de nervures déformé, de telle sort que la déformation du maillage suit la déformation du squelette de nervures.

Les études sur la courbure des feuilles des plantes du point de vue biophysique ont soulevé la question de savoir quel rôle, le cas échéant, les gènes jouent dans le contrôle de la courbure des feuilles. Nath et al. [UWRE03] ont proposé une méthode qui étudie le mutant Antirrhinum pour simuler la croissance de feuille en prenant en compte la croissance excessive dans les régions marginales. Il y a aussi des chercheurs qui étudient le modèle de l'agité ou de froissement dans les feuilles en utilisant l'analyse physique, par exemple Sharon et al. [EBL07].

Conclusion

Quelques méthodes sont proposées pour modéliser les feuilles. Les plupart de ces méthodes créent des feuilles en utilisant une image de feuille réelle. Ces méthodes génèrent une feuille en extrayant la bordure de l'image de feuille. La création du système de nervure est aussi traité par l'extraction du squelette de la bordure de feuille. Des caractéristiques biologiques et biophysiques sont utilisées afin d'obtenir des résultats plus réalistes. Cependant dans la nature, chaque type de feuille présente différentes formes de cette feuille, ce qui rend la génération de variété de feuilles très importante, point qui est pas largement négligé par ces méthodes.

1.3 Conclusion

Nous avons abordé les travaux sur la génération et le contrôle de formes naturelles en présentant et résumant leurs avantages et inconvénients. Deux catégorie de méthodes sont présentées : la génération procédurale et les méthodes d'éditions. Nous n'avons pas considéré les méthodes de simulation parce que nous pensons qu'elles ne présentent qu'un contrôle faible sur leur résultat final.

Les méthodes de génération procédurale sont très efficaces pour générer rapidement et automatiquement des formes naturelles artificielles à partir de très peu d'informations. De plus, la création de variétés d'objets se fait simplement en modifiant les différents paramètres. Toutefois, le nombre de paramètres à manipuler est souvent très grand, et le changement de la valeur d'un seul paramètre tend à modifier la forme globale de l'objet, et il devient difficile à l'utilisateur d'obtenir rapidement l'objet désiré. Le grand inconvénient de ces méthodes est le manque de contrôle sur le résultat final. Celui-ci a été le sujet de nombreuses recherches. Les méthodes qui adressent ce problème ne présentent pas généralement assez de contrôle sur la forme générée et le nombre de paramètres augmente beaucoup ce qui rend l'ajustement de ces paramètres très difficile pour un utilisateur non spécialisé. De plus, la structure de données nécessite d'être stockée ce qui rend le modèle non-procédural.

Les méthodes d'édition sont les plus utilisées quand le contrôle de la forme des objets est nécessaire. Quelques méthodes à base d'esquisses et d'exemples sont proposées pour la génération de terrain. Ces méthodes prouvent leur capacité et leur efficacité d'augmenter le réalisme dans leur résultats tout en présentant plus de contrôle sur ces résultats. De nombreuses méthodes à base d'esquisses et d'exemples sont utilisées pour la création de plantes ainsi que de feuilles. Ces méthodes augmentent le contrôle sur les modèles utilisés pour créer de tels objets (L-système, IFS, à base de botanique) tout en préservant le réalisme dans leurs résultats. Bien que les méthodes d'édition présentent plusieurs avantages en les utilisant pour la création de formes naturelles, ces méthodes ne sont pas génériques : il faut avoir un algorithme pour chaque type d'objet, par exemple l'algorithme utilisé pour créer un terrain à partir d'esquisses ne peut pas être utilisé pour créer des plantes. De plus ces méthodes ne peuvent pas créer une variété de formes même pour le même type d'objet.

Nous allons, à travers les chapitres suivants, présenter trois différentes approches pour créer une variété de feuilles ainsi que contrôler la génération de terrain. Les résultats obtenus sont convaincants et réalistes.

Chapitre 2

Modèle basé sur les IFS

Sommaire

Le modèle de base sur lequel repose celui que nous allons utiliser a été développé dans notre laboratoire. Il a fait l'objet de la thèse de Chems Eddine Zair [Zai98]. Il repose sur la fusion de deux modèles : les IFS, modèle fractal bien connu, et les formes à pôles, formalisme très utilisé en CAO[3]. Le résultat est appelé IFS projeté.

Dans un premier temps, nous introduisons les IFS sans rentrer trop dans les détails pour lesquels le lecteur pourra se reporter aux études de Barnsley [Bar88, Bar93]. Dans la suite, nous explicitons le modèle utilisé. Pour cela, nous exposons les procédés de paramétrisation des attracteurs d'IFS en présentant les fonctions d'adressage, puis la notion d'attracteur d'IFS projeté. Nous introduisons aussi l'approximation des courbes ou des surface en utilisant les IFS projetés qui était l'objet de la thèse d'Eric Guérin [Gué02]. Une méthode efficace pour la construction d'IFS projeté qui est présentée dans son travail, sera utilisée pour introduire comment il est possible d'ajouter la notion de détail dans les IFS projetés.

2.1 Modèle fractal

2.1.1 IFS : Iterated Function System

Fondés sur un formalisme mathématique rigoureux, les IFS sont un outil puissant pour l'analyse comme pour la synthèse d'objets fractals. Hutchinson [Hut81] fut le premier à les étudier en appliquant le théorème du point fixe aux ensembles de compacts d'un espace métrique complet. Barnsley [Bar88] a développé ce formalisme et l'a utilisé dans toute une série d'applications, entre autres, en infographie et en compression d'image. Dans cette section, nous allons rappeler la construction de ce modèle.

Théorème du point fixe

Le théorème du point fixe nécessite la définition d'un espace métrique complet ainsi que d'une application contractante.

Définition 1 ***(espace métrique)*** *On appelle espace métrique tout ensemble $\mathcal{X}$ muni d'une distance d. On dira que $\mathcal{X}$ est métrique complet si toute suite de Cauchy définie dans $\mathcal{X}$ admet une limite dans $\mathcal{X}$.*

Définition 2 ***(application contractante)*** *Soient $(\mathcal{X}, d)$ un espace métrique complet et*

[3]Conception assistée par ordinateur

T une application de $\mathcal{X}$ dans $\mathcal{X}$. L'application T est dite contractante si :

$$\exists s < 1, \forall p_1, p_2 \in \mathcal{X}, d(T(p_1), T(p_2)) < sd(p_1, p_2)$$

Théorème 1 ***(Théorème du point fixe)*** *Soit $(\mathcal{X}, d)$ un espace métrique complet. Soit $T : \mathcal{X} \mapsto \mathcal{X}$ une application contractante. Alors il existe un point unique $c \in \mathcal{X}$, appelé le point fixe de T, tel que $T(c) = c$.*

Pour une plus grande simplicité dans la suite, on notera T comme un opérateur $T\lambda = T(\lambda)$.

Espaces de compacts, opérateur de Hutchinson

La théorie des IFS repose sur la définition, à partir d'un espace métrique complet $(\mathcal{X}, d)$, d'un autre espace métrique complet, qui contient tous les compacts du premier. Nous noterons $\mathcal{H}(\mathcal{X})$ l'ensemble des compacts non vides de $(\mathcal{X}, d)$. La distance utilisée pour $\mathcal{H}(\mathcal{X})$ est construite grâce à d, elle est appelée distance de Hausdorff et est notée d_H.

Définition 3 ***(distance de Hausdorff)*** *Soient A et B deux compacts non vides d'un espace métrique $\mathcal{X}$. La distance de Hausdorff de A à B est définie par :*

$$d_H(A, B) = \max \left\{ \max_{p \in A} \min_{q \in B} d(p, q), \max_{p \in B} \min_{q \in A} d(p, q) \right\}$$

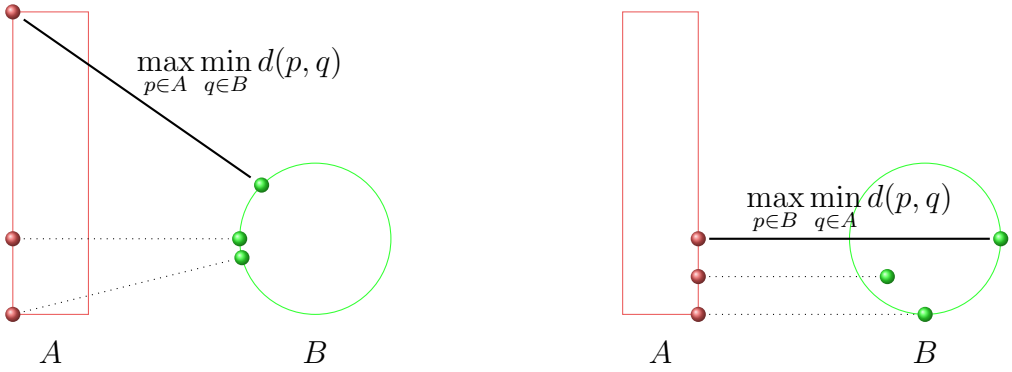
De façon plus parlante, $\max_{p \in A} \min_{q \in B} d(p, q)$ représente le maximum de la distance entre un point de A et le compact B. De la même façon, $\max_{p \in B} \min_{q \in A} d(p, q)$ représente le maximum de la distance entre un point de B et le compact A. La figure 3 donne un exemple de calcul de distance de Hausdorff. On comprend vite par cet exemple que la distance de Hausdorff fait intervenir des paramètres comme la forme et la disposition des compacts considérés, et non uniquement un éloignement moyen. Il s'agit donc d'une distance « visuelle ».

Le théorème suivant est la base de la théorie des IFS [Bar88].

Théorème 2 *Si $(\mathcal{X}, d)$ est un espace métrique complet, alors $(\mathcal{H}(\mathcal{X}), d_H)$ l'est aussi.*

Des applications contractantes dans $(\mathcal{H}(\mathcal{X}), d_H)$ peuvent être définies à partir d'applications contractantes dans $(\mathcal{X}, d)$. C'est le cas de l'opérateur de Hutchinson. Nous noterons l'action d'un opérateur sur un compact de la façon suivante :

$$TK = \bigcup_{p \in K} \{Tp\}$$



(a) Calcul du premier terme (b) Calcul du second terme

FIG. 3 – Exemple de distance de Hausdorff entre deux compacts de $(\mathbb{R}, d)$

Proposition 1 ***(opérateur de Hutchinson)*** *Soient N opérateurs continus $T_i : \mathcal{X} \to \mathcal{X}, i = 0, \ldots, N-1$, on peut leur associer l'opérateur continu défini par :*

$$K \in \mathcal{H}(\mathcal{X}) \mapsto \mathcal{T}K \in \mathcal{H}(\mathcal{X}) = \bigcup_{i=0}^{N-1} T_i K$$

$\mathcal{H}(\mathcal{X})$ est appelé opérateur de Hutchinson associé à $\mathcal{T}$. De plus, les propriétés de contraction de l'opérateur de Hutchinson sont liées à celles des T_i.

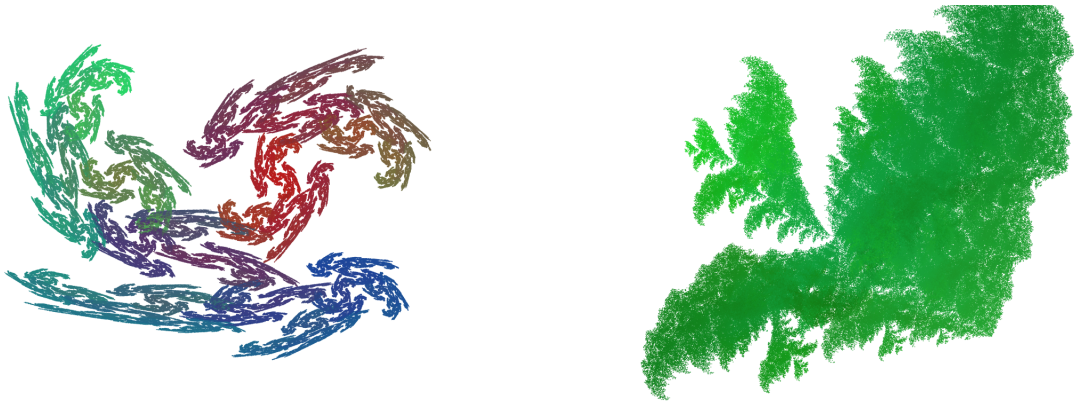
Proposition 2 *Si tous les opérateurs $T_0, \ldots, T_{N-1}$ sont contractants de constantes de contraction $s_0, \ldots, s_{N-1}$ alors l'opérateur de Hutchinson associé à $\mathbb{T} = \{T_0, \ldots, T_{N-1}\}$ est contractant pour la distance de Hausdorff et de constante de contraction $s = \max\{s_0, \ldots, s_{N-1}\}$.*

2.1.2 Définition d'un IFS

Définition 4 ***(IFS)*** *Soit $(\mathcal{X}, d)$ un espace métrique complet. Nous appelons IFS tout ensemble fini $\mathbb{T} = \{T_0, \ldots, T_{N-1}\}$ d'opérateurs contractants sur $\mathcal{X}$.*

Les propositions précédentes permettent d'associer à cet ensemble d'opérateurs un opérateur de Hutchinson qui est contractant dans l'espace métrique complet $(\mathcal{H}(\mathcal{X}), d_H)$. Il est donc possible d'appliquer le théorème du point fixe.

Théorème 3 ***(théorème d'existence d'attracteur)*** *Pour tout IFS $\mathbb{T}$, il existe un com-*



(a) (b)

FIG. 4 – Exemples d'attracteurs d'IFS dans le plan.

pact unique non vide A de $\mathcal{H}(\mathcal{X})$ tel que :

$$\begin{aligned} A &= \mathbb{T}A \\ &= T_0A \cup \ldots \cup T_{N-1}A \end{aligned}$$

A est appelé attracteur de $\mathcal{T}$ et sera noté $\mathcal{A}(\mathbb{T})$.

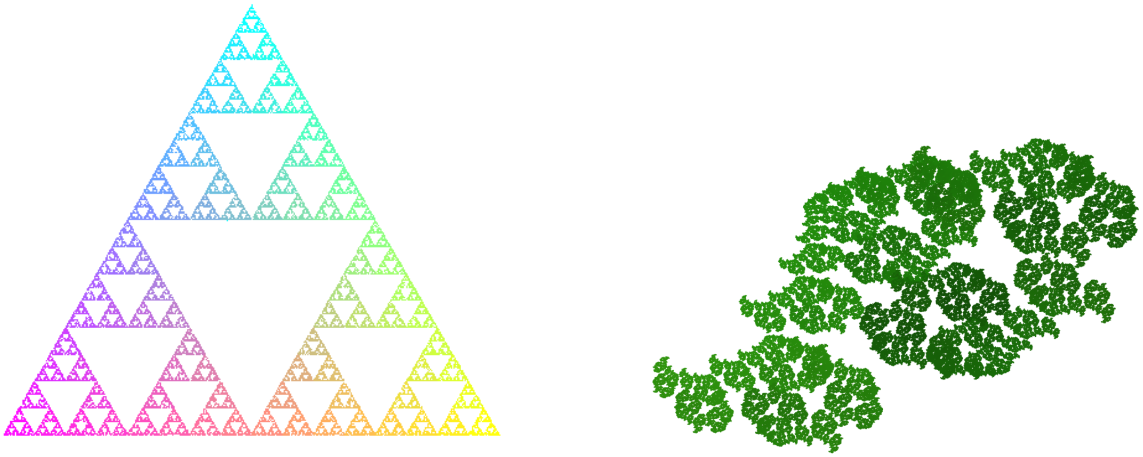
Il est donc possible d'associer à tout IFS $\mathbb{T}$ un compact, appelé attracteur. Par définition, tout attracteur possède la propriété d'auto-similarité au sens large, c'est-à-dire :

$$A = T_0A \cup \ldots \cup T_{N-1}A$$

avec les T_i appartenant à une classe de fonctions contractantes.

Exemples

Nous prenons le cas où $\mathcal{X} = \mathbb{R}^2$, les compacts dans ce cas peuvent être assimilés à une figure du plan. Les figures 4 et 5 montrent quatre attracteurs d'IFS dont les transformations sont affines. La propriété d'autosimilarité est plus ou moins perceptible visuellement. Dans la figure 4 elle est difficilement décelable tandis que dans les figures 5, elle saute aux yeux.



(a) Le classique triangle de Sierpinsky (b)

FIG. 5 – Autres exemples d'attracteurs d'IFS dans le plan.

2.1.3 Paramtrisation d'attracteurs

Dans le paragraphe précèdent, les attracteurs d'IFS sont définis comme des ensembles de points. Il est possible cependant d'en faire des modèles paramétrés. Pour cela, il faut numéroter les opérateurs de l'IFS, afin de construire une fonction d'adressage[Bar88, Edg90]. Dans le cas où l'attracteur est une courbe ou une surface, cette paramétrisation formelle peut être mise en correspondance avec une paramétrisation numérique.

Fonction d'Adressage

Définition 5 ***(alphabet)*** *Soit un IFS* $\mathbb{T} = \{T_0, \ldots, T_{N-1}\}.\Sigma = \{0, ..., N-1\}$ *sera appelé l'alphabet associé à* $\mathbb{T}$.

En notant Σ^* l'ensemble des mots finis sur Σ, et Σ^ω l'ensemble des mots infinis sur Σ, on a le résultat suivant (voir Barnsley[Bar88]) :

Théorème 4 *Soit* $\theta = \theta_1\theta_2\theta_3 \ldots \in \Sigma^\omega$. *Pour tout* $\lambda \in \mathcal{X}$ *la limite :*

$$\lim_{j\to\infty} T_{\theta 1} \ldots T_{\theta j}\lambda$$

existe et est indépendante de λ.

On peut définir une fonction d'adressage, qui, à un mot infini sur Σ, associe un point de l'attracteur.

Définition 6 ***(fonction d'adressage)*** *Soient $\mathbb{T}$ un IFS et Σ son alphabet associé. On appelle fonction d'adressage la fonction ϕ qui à un mot infini de Σ fait correspondre un élément de $\mathcal{X}$ de la façon suivante :*

$$\begin{aligned} \phi : \Sigma^w &\mapsto \mathcal{X} \\ \theta &\mapsto \phi(\theta) = \lim_{j\to\infty} T_{\theta 1} \dots T_{\theta j} \lambda \end{aligned}$$

avec $\lambda \in \mathcal{X}$ quelconque.

Formes paramétrées

Cette fonction d'adressage est insuffisante pour décrire une forme paramétrée de type courbe ou surface. Dans ces cas, il est plus pratique de disposer d'une fonction dont les arguments sont des scalaires $s \in [0,\, 1]$. Sous certaines conditions sur l'IFS $\mathbb{T}$, que nous verrons dans la suite, une telle fonction existe et definit un arc de courbe ou un carreau de surface. Pour cela, nous utilisons le développement de s en base N :

$$s = \sum_{i=1}^{\infty} \frac{1}{N^i} \theta_i$$

Définition 7 ***(fonction de paramétrage d'un IFS)*** *Soient $\mathbb{T}$ un IFS, Σ son alphabet, et ϕ sa fonction d'adressage. La fonction de parametrage Φ de l'IFS $\mathbb{T}$ est définie par :*

$$\begin{aligned} \Phi : [0,\, 1] &\mapsto \mathcal{X} \\ s &\mapsto \Phi(s) = \phi(\theta) \end{aligned}$$

avec $s = \sum_{i=1}^{\infty} \frac{1}{N^i} \theta_i$.

Cette fonction de paramétrage à un seul paramètre permet de décrire des objets tels que les courbes.

Nous pouvons définir de manière similaire une fonction de paramétrage à deux paramètres [Gué02] qui permet de décrire des objets tels que les surfaces. Cette fonction est définie par :

$$\begin{aligned} \Phi : [0,\, 1]^2 &\mapsto \mathcal{X} \\ s &\mapsto \Phi(s,t) = \phi(\theta) \end{aligned}$$

avec $s = \sum_{i=1}^{\infty} \frac{1}{N^i} \sigma_i$, $t = \sum_{i=1}^{\infty} \frac{1}{N^i} \rho_i$ et $\theta = \theta_1 \dots \theta_n \dots = (\sigma_1, \rho_1) \dots (\sigma_n, \rho_n) \dots$.

Remarque : Nous présentons la fonction de paramétrage pour expliquer la similarité de définition entre les formes à pôle et les formes fractales à pôle. Dans notre travail nous n'utilisons que la fonction d'adressage.

2.1.4 Formes fractales à pôles

Le modèle IFS est un outil puissant pour la création de formes fractales, en particulier dans le domaine artistique. Cependant, il est insuffisant dans un contexte de modélisation géométrique. Le problème apparaît lorsque l'on veut pouvoir déformer l'attracteur, action courante en modélisation géométrique. L'idée des attracteurs d'IFS projetés a vu le jour dans notre laboratoire. Elle a été développée dans la thèse de Chems Eddine Zair [Zai98]. Le principe est de fusionner deux modèles : les IFS et les formes à pôles. Dans un premier temps, nous allons rappeler le formalisme des formes à pôles. Puis, nous verrons comment il est possible de l'étendre aux IFS. Enfin, nous montrerons des exemples de courbes et surfaces fractales qui sont des attracteurs d'IFS projetés.

Formes à pôles

L'idée principale des formes à pôles est de séparer la fonction qui représente une courbe ou une surface en deux parties : le polygone de contrôle $(p_j)_{j\in J}$ et les fonctions de mélange $(\Phi_j)_{j\in J}$.

La formule qui décrit une courbe est la suivante :

$$F(s) = \sum_{j\in J} \Phi_j(s) p_j$$

Le polygone de contrôle sert à contrôler la forme globale de la courbe à modéliser. Il permet de déformer celle-ci d'une manière intuitive. Il est noté $P = (p_j)_{j\in J}$ où J désigne l'ensemble des indices des points de contrôle. L'ensemble des indices pour les courbes est $J = \{0, \ldots, m\}$ de manière à décrire un polygone de contrôle. Les fonctions de mélange quant à elles indiquent la façon dont les points de contrôle sont « mélangés » pour constituer la forme finale. Elles sont notées $(\Phi_j)_{j\in J}$. En modélisation géométrique classique, les fonctions de mélange sont fixées une fois pour toutes et l'utilisateur dispose des points de contrôle pour modifier la forme. Ces fonctions de mélange sont des polynômes, des B-splines, des NURBS,... et possèdent la propriété suivante :

$$\forall s \in [0,\, 1],\ \sum_{j\in J} \Phi_j(s) = 1$$

Elles permettent ainsi de combiner les points de contrôle. La figure 6 montre un exemple de modélisation géométrique grâce à une courbe B-spline.

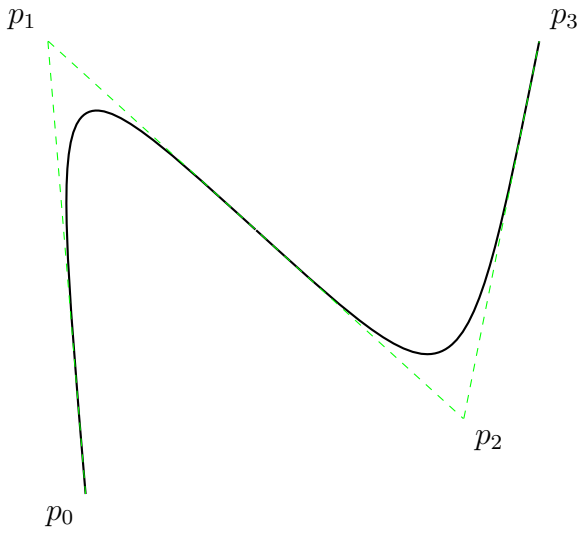


FIG. 6 – Courbe B-spline

Remarque : Les points de contrôle p_j sont choisis dans un espace dit de modélisation. En pratique ce sera $\mathbb{R}^2$ ou $\mathbb{R}^3$.

Cette notion s'étend aux surfaces grâce à un ensemble d'indices $J = \{0, \ldots, n\} \times \{0, \ldots, m\}$. La formule qui décrit la surface est alors :

$$F(s,t) = \sum_{j \in J} \Phi_j(s,t) p_j$$

et la propriété des fonctions de mélange se transpose ainsi :

$$\forall (s,t) \in [0,\, 1]^2 \,, \sum_{j \in J} \Phi_j(s,t) = 1$$

L'idée des formes fractales à pôles est de fabriquer des jeux de fonctions de mélange différents des fonctions standards et ayant des propriétés fractales.

IFS projeté

Pour fabriquer des fonctions de mélange fractales, il faut les définir en coordonnées barycentriques. Il faut donc choisir comme espace d'itération un espace barycentrique.

Définition 8 ***(espace barycentrique)*** *Soit J un ensemble d'indices. L'espace barycen-*

trique $\mathcal{B}^J$ associé à J est défini par :

$$\mathcal{B}^J = \left\{ (\lambda_j)_{j\in J} | \sum_{j\in J} \lambda_j = 1 \right\}$$

avec $\lambda_j \in \mathcal{R}$.

Il faut maintenant se fixer un semigroupe d'itérations qui opère dans cet espace barycentrique. La solution la plus simple est d'utiliser des matrices dont les colonnes sont barycentriques. Dans toute la suite de ce chapitre, nous appelons ces matrices matrices de subdivision. En effet, les matrices composant l'IFS correspondent aux matrices utilisées dans plusieurs schémas de subdivision de courbes ou de surfaces (algorithme de De Casteljau, Overveld, Daubechies,...).

Définition 9 ***(semigroupe de matrices barycentriques)*** *Soit J un ensemble d'index. Le semigroupe de matrices barycentriques S_J associé à J est défini par :*

$$S_J = \left\{ T | \sum_{j\in J} T_{i,j} = 1, \forall i \in J \right\}$$

Remarque : Nous constaterons facilement qu'un espace barycentrique muni de la distance euclidienne est complet[ZT96]. En effet, il s'agit d'un hyperplan, équivalant métriquement à $\mathbb{R}^{card(J)-1}$.

Grâce à ces choix, il est possible de « projeter » un attracteur d'IFS.

Définition 10 ***(attracteur d'IFS projeté)*** *Soient J un ensemble d'indices, $\mathcal{T}$ un IFS constitué d'opérateurs de S_J, et $\mathcal{P} = (p_j)_{j\in J}$ une famille de points de contrôle. L'attracteur d'IFS projeté associé à $\mathcal{T}$ est défini par :*

$$\begin{aligned} \mathcal{PA}(\mathcal{T}) &= \{\mathcal{P}\lambda | \lambda \in \mathcal{A}(\mathcal{T})\} \\ &= \mathcal{P}\phi(\theta) \\ &= \sum_{j\in J} \phi_j(\theta) p_j \end{aligned}$$

où $\mathcal{P}\lambda = \sum_{j\in J} \lambda_j p_j$

Ceci est une définition ensembliste de l'attracteur d'IFS projeté. Il est en fait possible de définir celui-ci de manière fonctionnelle (ou paramétrique) grâce à l'équation de paramé-

trisation de l'IFS :

$$F(s) = P\Phi(s) = \sum_{j \in J} \Phi_j(s) p_j$$

ou avec 2 paramètres dans le cas des surfaces :

$$F(s,t) = P\Phi(s,t) = \sum_{j \in J} \Phi_j(s,t) p_j$$

Exemples

Nous allons ici donner des exemples d'attracteurs d'IFS projetés, pour illustrer la variété des formes qui peut être générée grâce à ce modèle.

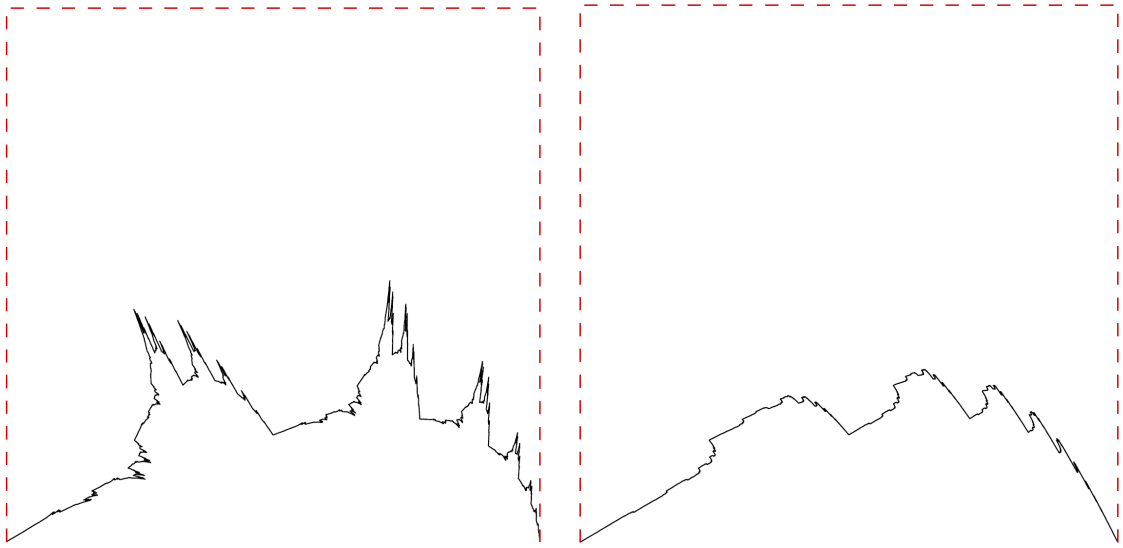
Courbes

Les figures 7 et 8 sont des exemples de courbes fractales à pôles. Elles ont toutes été générées avec le même polygone de contrôle disposé en carré, et le même nombre de transformations : deux matrices de subdivision T_0 et T_1. En jouant sur les coefficients contenus dans les matrices de subdivision, nous pouvons obtenir des courbes différentes. Lorsqu'il existe une symétrie entre les deux matrices, on obtient une courbe symétrique [Zai98] (Figure 8).

La figure 9 illustre les fonctions de mélange Φ, et leur équivalent fractal. La figure 9 (gauche) montre un exemple de fonction de mélange (pour une B-spline) en explicitant chacune des quatre composantes associées aux quatre points de contrôle. La figure 9 (droite) illustre un cas fractal, avec la même représentation. Dans les deux figures, l'abscisse représente la valeur du paramètre de la courbe. Le formalisme est donc le même dans les deux cas, mais on a remplacé des fonctions à base de polynômes par des fonctions fractales définies par un modèle IFS.

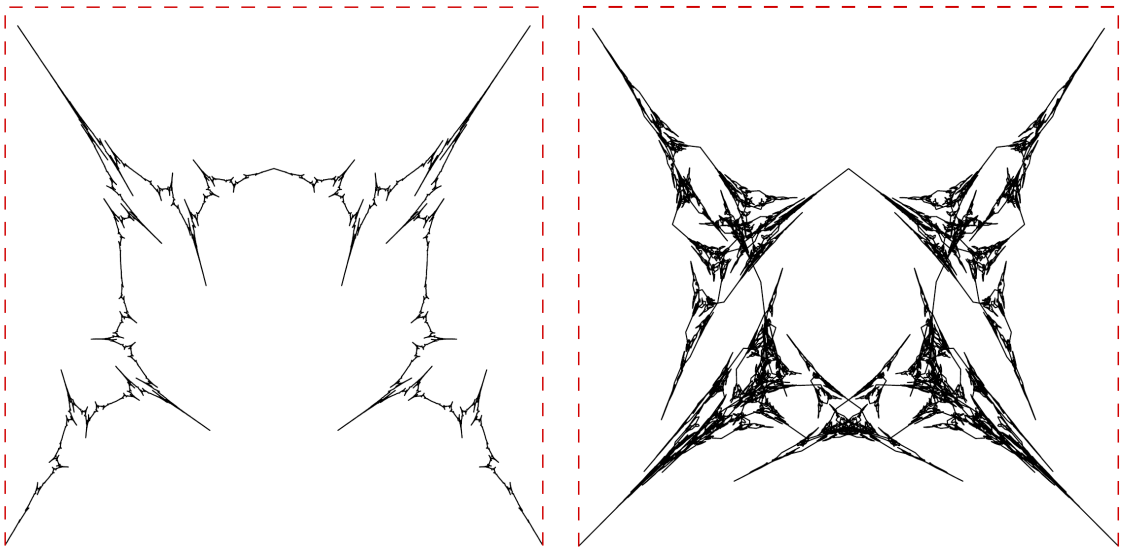
Surfaces

La figure 10 montre un exemple de surface fractale à pôles. Dans cet exemple, une grille de 5×5 points de contrôle a été utilisée.



(a) (b)

FIG. 7 – Courbes faiblement accidentées



(a) (b)

FIG. 8 – Courbes symétriques

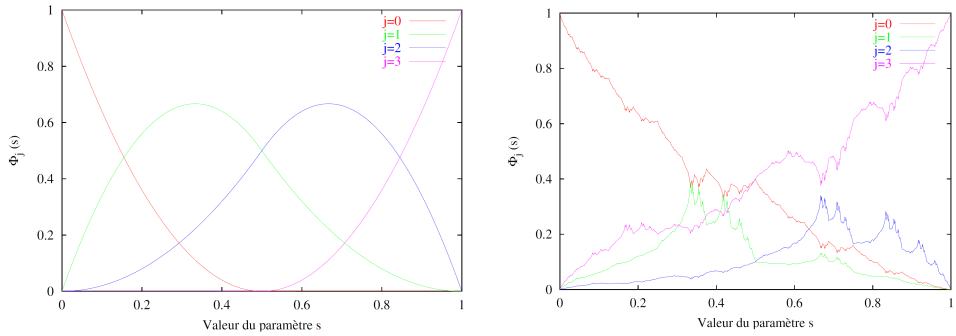


FIG. 9 – Exemple de fonction de mélange d'une B-spline (gauche), et d'une courbe fractale (droite)

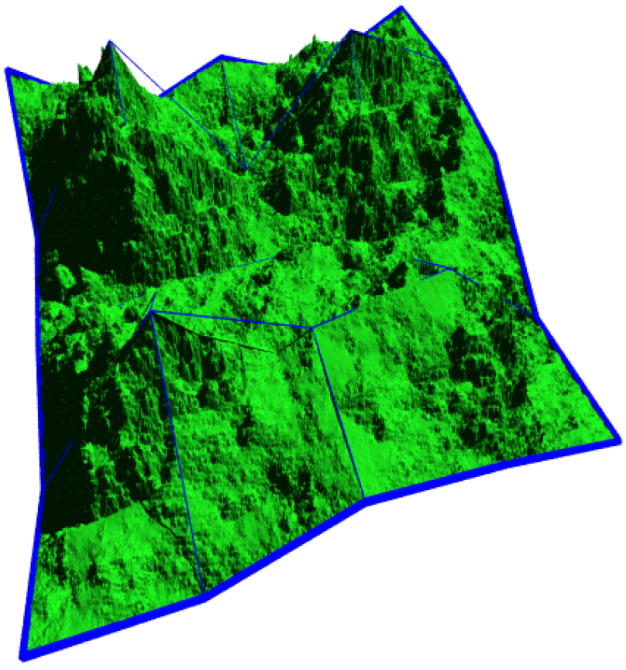


FIG. 10 – Exemple de surface fractale à pôles avec une grille de contrôle de 5×5 points de contrôle

2.2 Insertion de détail

Nous avons présenté dans les sections précédentes de ce chapitre, la théorie des IFS et son extension vers les formes à pôle appelée IFS projeté. Dans cette section nous introduisons notre contribution pour ce chapitre en présentant comment nous étendons les IFS projetés avec la notion de détail. Ce nouveau modèle est utilisé pour représenter les courbes ou les surfaces avec un minimum d'informations. Pour cette raison une étape d'optimisation est appliquée afin d'approximer le modèle par des courbes ou des surfaces.

2.2.1 Visualisation

Afin de présenter une formule qui se rapproche de notre formule générale (1) page (2), nous allons commencer par présenter un moyen de visualiser les IFS projetés [Gué02]. Le procédé de visualisation se rapproche de celui de la construction des courbes de Bézier par l'algorithme de De Casteljau.

À chaque IFS $\mathbb{T} = \{T_0, \ldots, T_{N-1}\}$ correspond un compact unique $\mathcal{A}(\mathbb{T})$ appelé attracteur. La méthode de construction de cet attracteur se déduit du théorème du point fixe. Le principe est de construire une suite qui converge vers l'attracteur

$$\lim_{n\to\infty} K_n = \mathcal{A}(\mathbb{T})$$

Le résultat général suivant permet de construire une telle suite :

Proposition 3 *Soient $(\mathcal{X}, d)$ un espace métrique complet et T une application contractante sur $\mathcal{X}$. On a :*

$$\forall \lambda \in \mathcal{X},\ \lim_{n\to\infty} T_n\lambda = c$$

où c désigne le point fixe de T. Autrement dit, toute suite définie par :

$$(a_n)_{n\in\mathbb{N}} = \begin{cases} a_0 & = \lambda \in \mathcal{X} \\ a_{n+1} & = Ta_n,\ \forall n \geq 0 \end{cases}$$

converge vers c.

Pour appliquer ce procédé aux IFS, il suffit d'itérer l'opérateur de Hutchinson. On obtient le théorème fondamental de visualisation déterministe [Gué02] :

Théorème 5 ***(visualisation déterministe des IFS)*** *Soient $(\mathcal{X}, d)$ un espace métrique*

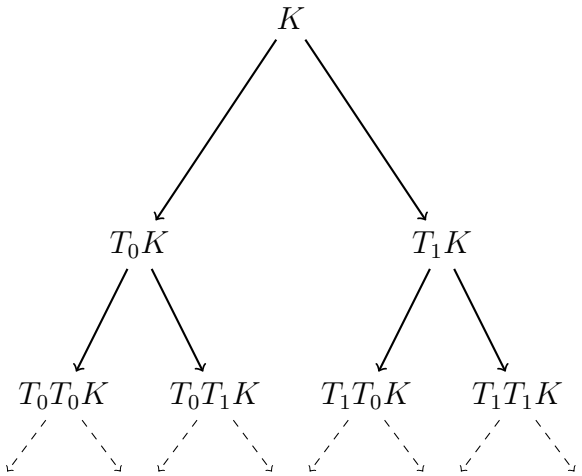


FIG. 11 – Principe de visualisation d'un IFS

complet et $\mathbb{T}$ *un IFS. Toute suite définie par :*

$$(K_n)_{n\in\mathbb{N}} = \begin{cases} K_0 & = K \in \mathcal{H}(\mathcal{X}) \\ K_{n+1} & = \mathbb{T}K_n,\ \forall n \geq 0 \end{cases}$$

converge vers l'attracteur d'IFS $\mathcal{A}(\mathbb{T})$*.*

Le procédé est donc simple, n'importe quel compact de départ produira le même résultat. En pratique, on prend un compact pour lequel les transformations sont faciles à calculer et qui présente un bon rendu visuel, une sphère par exemple pour $\mathcal{X} = \mathbb{R}^3$. Ce principe s'apparente à la construction récursive d'un arbre. La figure 11 montre l'exemple d'un arbre associé à la visualisation d'un IFS.

Prenons maintenant le cas des attracteurs d'IFS projetés. Nous expliquons ici le cas des courbes, mais il suffit de remplacer le mot polygone par grille pour l'étendre aux surfaces. Un attracteur projeté est décrit par :

$$\begin{aligned} F(s) &= P\Phi(s) \\ &= P\phi(\theta) \\ &= P \lim_{j\to\infty} T_{\theta 1} \dots T_{\theta j}\lambda \end{aligned}$$

Or l'action de P est affine, on peut donc le faire entrer dans la limite :

$$F(s) = \lim_{j\to\infty} PT_{\theta 1} \dots T_{\theta j}\lambda$$

Un procédé de construction des attracteurs projetés peut être déduit de cette formule, ce

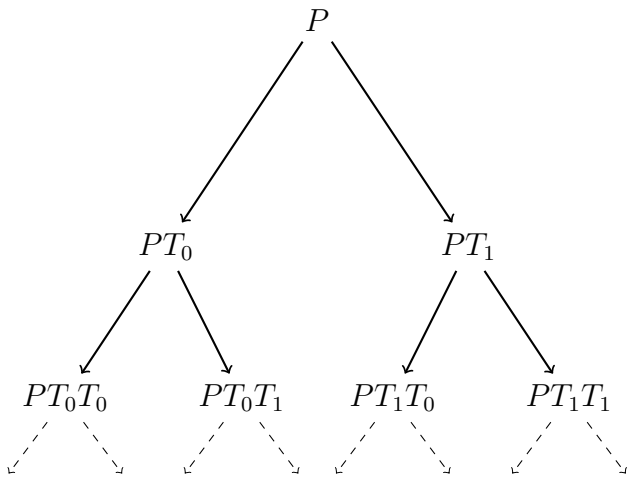


FIG. 12 – Arbre de construction d'un IFS projeté

procédé consiste à construire une suite d'ensembles de polygones en appliquant l'IFS à droite, directement sur les points de contrôle[Gué02] :

Proposition 4 ***(visualisation d'attracteurs d'IFS projetés)*** *Soient $(\mathcal{X}, d)$ un espace métrique complet et $\mathbb{T}$ un IFS, et P un vecteur de points de contrôle. Soit la suite définie par :*

$$(S_n)_{n\in\mathbb{N}} = \begin{cases} S_0 & = \{\mathcal{P}\} \\ S_{n+1} & = S_n\mathbb{T}, \forall n \geq 0 \end{cases} \tag{2.1}$$

Alors, on a :

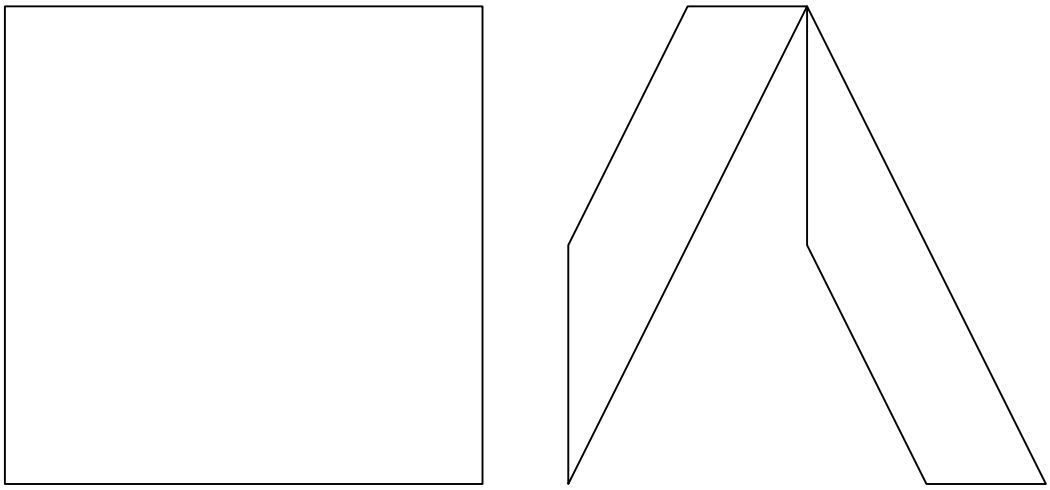
$$\lim_{n\to\infty} S_nK = P\mathcal{A}(\mathbb{T})$$

avec $K \in \mathcal{H}(\mathcal{X})$.

Remarque : S_n est un ensemble fini de polygones que l'on peut développer de manière récursive grâce à un arbre (Figure 12) :

$$S_n = P\mathbb{T}^n = \{PT_{\theta 1} \dots PT_{\theta n} \,|\, |\theta| = n\}$$

La figure 13 illustre le processus de construction d'attracteurs d'IFS projetés en montrant les différents polygones $PT_{\theta 1} \dots PT_{\theta n}$ composant S_n des 2 premières itérations (Figures 13(a) à 13(c)) et après 12 itérations (Figure 13(d)).



(a) Polygone de contrôle initial $S_0 = P$

(b) Après une itération $S_1 = P\mathbb{T}$

(c) Après 2 itérations $S_2 = P\mathbb{T}^2$

(d) Après 12 itérations $S_{12} = P\mathbb{T}^{12}$

FIG. 13 – Les étapes récursives de construction d'une courbe fractale à pôles

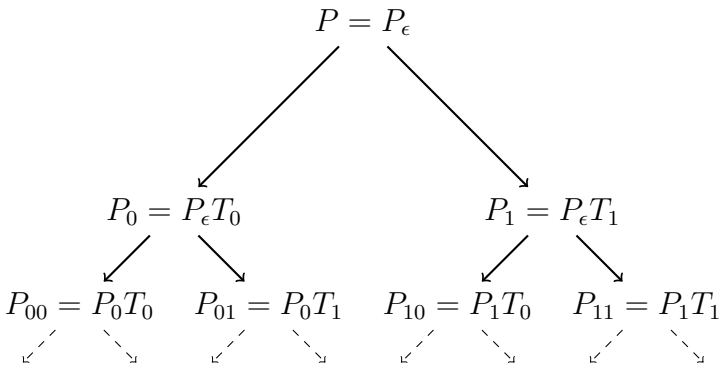


FIG. 14 – Arbre de construction d'un IFS projeté avec $\Sigma = \{0, 1\}$

2.2.2 Ajout d'une partie de détail

Prenons l'équation (2.1) page (42) qui dit que : S_n est un ensemble fini de polygones que nous pouvons développer de manière récursive grâce à un arbre (Figure 12 page 42)

$$S_n = P\mathbb{T}^n = \{PT_{\theta 1} \dots PT_{\theta n} \,|\, |\theta| = n\}$$

Nous voulons maintenant extraire une formule de cette formule que nous pouvons représenter en utilisant des mots de Σ où Σ est l'alphabet associé à $\mathbb{T}$. Notons $T_\theta = T_{\theta_1} \dots T_{\theta_n}$ et $P_\theta = PT_\theta$. Nous pouvons écrire :

Proposition 5 ***(autre forme de visualisation d'attracteurs d'IFS projetés)*** *Soient $(\mathcal{X}, d)$ un espace métrique complet et $\mathbb{T}$ un IFS, Σ l'alphabet associé à $\mathbb{T}$,et P un vecteur de points de contrôle. La suite définie par l'équation (2.1) page (42) est équivalente à :*

$$\begin{aligned} P_{\theta i} &= PT_\theta T_i \\ &= P_\theta T_i \end{aligned} \tag{2.2}$$

où $i \in \Sigma$ et θ,$\theta i \in \Sigma^\omega$. Nous notons le vecteur de points de contrôle $P = P_\epsilon$ où ϵ représente le mot vide.

Selon cette formule l'arbre qui représente la construction d'attracteurs d'IFS projetés (Figure 12 page 42) peut être représenté comme sur la figure 14.

Inspiré par les travaux de Tosan et al.[TBSS+07] nous pouvons ajouter une partie de détail sur la formule (2.2) de cette manière :

$$P_{\theta i} = P_\theta T_i + \delta P_\theta U_i \quad \text{où} \quad i \in \Sigma \tag{2.3}$$

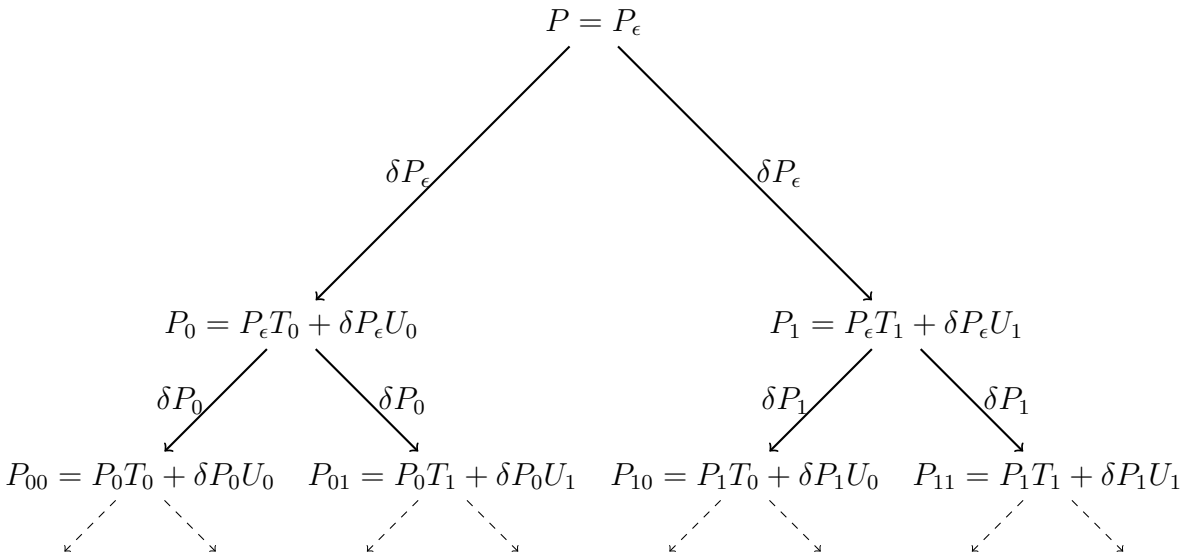


FIG. 15 – Arbre de construction d'un IFS projeté équipé de détail avec $\Sigma = \{0, 1\}$

Où $\delta\mathcal{P}_\theta U_i$ est une liste ordonnée de vecteurs de déplacement obtenue par la multiplication du vecteur de détail δP_θ avec la matrice U_i que nous appelons la matrice de déplacement de détail. Figure 15 représente l'arbre de construction de ce modèle.

Afin de bien définir l'addition (+) entre un polygone et une liste ordonnée de vecteurs nous introduisons la définition suivante :

Définition 11 ***(Déplacement d'un polygone avec une liste ordonnée de vecteurs)*** *Soient P un polygone et Q une liste ordonnée de vecteurs tels que le nombre de sommets n de P est égal au nombre de vecteurs de Q, nous définissons $P + Q$ comme étant un nouveau polygone ayant n points et nous le calculons comme ceci :*

$$P + Q = \{P_i + Q_i\}_{\{i=0\ldots n-1\}}$$

où $P_i + Q_i$ est un point dont les coordonnées sont la somme de chacune des coordonnées de P_i et Q_i.(Figure 16)

Maintenant nous définissons $\mathcal{P}^{J+1}$, $\mathcal{P}^J$, $\varphi(\mathcal{P}^J)$, $\psi(\mathcal{P}^J, \delta\mathcal{P}^J)$ et $\oplus$ dans l'équation (1) pour ce modèle.

Nous posons $\mathcal{P}^J = \{\mathcal{P}_\theta\}_{|\theta|=J \text{ et } \theta\in\Sigma^\star}$, alors $\mathcal{P}^{J+1} = \{\mathcal{P}_{\theta i}\}_{|\theta|=J \text{ et } i\in\Sigma \text{ et } \theta\in\Sigma^\star}$. Donc ici nous considérons que $\mathcal{P}^J$ représente un ensemble ordonné de polygones par exemple si $J = 2$ et $\Sigma = \{0, 1\}$ alors $\mathcal{P}^2 = (\mathcal{P}_\theta)_{\{|\theta|=2\}} = (\mathcal{P}_{00}, \mathcal{P}_{01}, \mathcal{P}_{10}, \mathcal{P}_{11})$.
$\varphi(\mathcal{P}^J)$ est défini par :

$$\varphi(\mathcal{P}^J) = \{\mathcal{P}_\theta T_i\}_{|\theta|=J \text{ et } i\in\Sigma \text{ et } \theta\in\Sigma^\star}$$

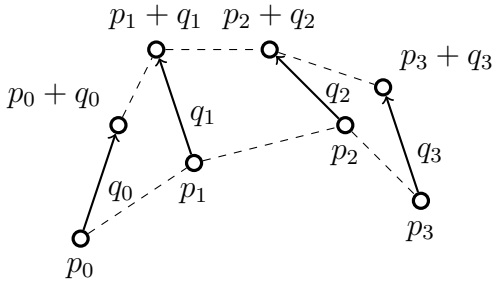


FIG. 16 – Déplacement d'un polygone.

et nous définissons $\psi(\mathcal{P}^J, \delta\mathcal{P}^J)$ par :

$$\psi(\mathcal{P}^J, \delta\mathcal{P}^J) = \{\delta\mathcal{P}_\theta U_i\}_{|\theta|=J \text{ et } i\in\Sigma \text{ et } \theta\in\Sigma^\star}$$

et finalement nous définissons $\oplus$ comme ceci :

Soit $(P_i)_{i=0,\ldots,m-1}$ un ensemble ordonné de m polygones, et $(Q_i)_{i=0,\ldots,m-1}$ un ensemble ordonné de m listes ordonnées de vecteurs, nous définissons $\oplus$ par :

$$(P_i)_{i=0,\ldots,m-1} \oplus (Q_i)_{i=0,\ldots,m-1} = (P_i + Q_i)_{i=0,\ldots,m-1}$$

où $P_i + Q_i$ est une somme entre un polygone et une liste ordonnée de vecteurs comme nous l'avons définie dans la définition (11).

Nous pouvons écrire :

$$(\mathcal{P}_{\theta i}) = (\mathcal{P}_\theta T_i) + (\delta\mathcal{P}_\theta U_i) \text{ où } |\theta| = J \text{ et } i \in \Sigma \text{ et } \theta \in \Sigma^\star$$

Si nous considérons un polygone de contrôle ayant n points , N transformations, dans l'espace $\mathbb{R}^d$, alors la dimension de la matrice T_i est $n \times n$, U_i est $n.(N-1) \times n$, $\mathcal{P}_\theta$ est $d \times n$, et $\delta\mathcal{P}_\theta$ est $d \times n.(N-1)$.

Le choix de la dimension des matrices de déplacement de détail et du vecteur de détail se base sur le fait que nous cherchons un système inversible.

En prenant la formule $\mathcal{P}_{\theta i} = \mathcal{P}_\theta T_i + \delta\mathcal{P}_\theta U_i$ pour tout $i \in \Sigma$ on a :

$$\begin{array}{ccc} \mathcal{P}_{\theta 0} & = & \mathcal{P}_\theta T_0 + \delta\mathcal{P}_\theta U_0 \\ \vdots & \vdots & \vdots \\ \mathcal{P}_{\theta N-1} & = & \mathcal{P}_\theta T_{N-1} + \delta\mathcal{P}_\theta U_{N-1} \end{array}$$

Maintenant nous pouvons écrire les formules précédentes sous forme matricielle :

$$(\mathcal{P}_{\theta 0}|\ldots|\mathcal{P}_{\theta N-1}) = (\mathcal{P}_\theta|\delta\mathcal{P}_\theta)\left(\begin{array}{c|c|c} T_0 & \ldots & T_{N-1} \\ \hline U_0 & \ldots & U_{N-1} \end{array}\right)$$

en utilisant la notation suivante :

$$R = \left(\begin{array}{c|c|c} T_0 & \ldots & T_{N-1} \\ \hline U_0 & \ldots & U_{N-1} \end{array}\right)$$

la formule devient :

$$(\mathcal{P}_{\theta 0}|\ldots|\mathcal{P}_{\theta N-1}) = (\mathcal{P}_\theta|\delta\mathcal{P}_\theta)\, R$$

la formule dite inverse est

$$(\mathcal{P}_\theta|\delta\mathcal{P}_\theta) = (\mathcal{P}_{\theta 0}|\ldots|\mathcal{P}_{\theta N-1})\, R^{-1}$$

ainsi R peut être apparenté à un filtre de synthèse et R^{-1} à un filtre d'analyse utilisés dans la transformée en ondelette (Figures 1 et 2).

Nous introduisons maintenant un exemple afin d'expliquer le modèle.

Exemple 2.3 *Soit* $\Sigma = \{0, 1\}$ *et nous employons l'espace* $\mathbb{R}^2$. *Soit* $\mathcal{P}_\epsilon = (p_0\ p_1\ p_2\ p_3)$ *nous avons alors* $\mathcal{P}_\epsilon = \begin{pmatrix} p_{0_x} & p_{1_x} & p_{2_x} & p_{3_x} \\ p_{0_y} & p_{1_y} & p_{2_y} & p_{3_y} \end{pmatrix}$, *dans ce cas nous avons deux transformations* (T_0, T_1) *dont la dimension est* 4×4, *et deux matrices de déplacement de détail* (U_0, U_1) *dont la dimension est* 4×4. *nous supposons que* (T_0, T_1) *et* (U_0, U_1) *sont comme dans la figure 17.*

$$T_0 = \begin{pmatrix} t0_{00} & t0_{01} & t0_{02} & t0_{03} \\ t0_{10} & t0_{11} & t0_{12} & t0_{13} \\ t0_{20} & t0_{21} & t0_{22} & t0_{23} \\ t0_{30} & t0_{31} & t0_{32} & t0_{33} \end{pmatrix} \quad \text{(a)} \qquad T_1 = \begin{pmatrix} t1_{00} & t1_{01} & t1_{02} & t1_{03} \\ t1_{10} & t1_{11} & t1_{12} & t1_{13} \\ t1_{20} & t1_{21} & t1_{22} & t1_{23} \\ t1_{30} & t1_{31} & t1_{32} & t1_{33} \end{pmatrix} \quad \text{(b)}$$

$$U_0 = \begin{pmatrix} u0_{00} & u0_{01} & u0_{02} & u0_{03} \\ u0_{10} & u0_{11} & u0_{12} & u0_{13} \\ u0_{20} & u0_{21} & u0_{22} & u0_{23} \\ u0_{30} & u0_{31} & u0_{32} & u0_{33} \end{pmatrix} \quad \text{(c)} \qquad U_1 = \begin{pmatrix} u1_{00} & u1_{01} & u1_{02} & u1_{03} \\ u1_{10} & u1_{11} & u1_{12} & u1_{13} \\ u1_{20} & u1_{21} & u1_{22} & u1_{23} \\ u1_{30} & u1_{31} & u1_{32} & u1_{33} \end{pmatrix} \quad \text{(d)}$$

FIG. 17 – Matrices de transformation et de déplacement de détail.

Le vecteur de détail est un vecteur de dimension 2×4 *:* $\delta\mathcal{P}_\epsilon = \begin{pmatrix} \delta p_{0_x} & \delta p_{1_x} & \delta p_{2_x} & \delta p_{3_x} \\ \delta p_{0_y} & \delta p_{1_y} & \delta p_{2_y} & \delta p_{3_y} \end{pmatrix}$.

La matrice de masque R dans ce cas est $R = \left(\begin{array}{c|c} T_0 & T_1 \\ \hline U_0 & U_1 \end{array}\right)$

Après avoir appliqué la première étape de synthèse, nous avons deux nouveaux polygones $(\mathcal{P}_0, \mathcal{P}_1)$ *que nous calculons comme ceci :*

$$\left(\mathcal{P}_0 \mid \mathcal{P}_1 \right) = \left(\begin{array}{cccc|cccc} p_{0_x} & p_{1_x} & p_{2_x} & p_{3_x} & \delta p_{0_x} & \delta p_{1_x} & \delta p_{2_x} & \delta p_{3_x} \\ p_{0_y} & p_{1_y} & p_{2_y} & p_{3_y} & \delta p_{0_y} & \delta p_{1_y} & \delta p_{2_y} & \delta p_{3_y} \end{array}\right) \left(\begin{array}{c|c} T_0 & T_1 \\ \hline U_0 & U_1 \end{array}\right)$$

Remarque : dans la deuxième étape de synthèse nous appliquons la matrice R deux fois une pour $(\mathcal{P}_0|\delta\mathcal{P}_0)$ et une autre pour $(\mathcal{P}_1|\delta\mathcal{P}_1)$. Puis dans la troisième étape nous l'appliquons quatre fois puis huit fois, *etc.*

Représentation de surface

De manière similaire aux courbes, une surface est considérée comme un ensemble de points $(p_{i,j})_{i,j=0\ldots n-1}$ placés sur une grille où n est la taille de cette grille. Nous définissions deux matrices R_l pour les lignes et R_c pour les colonnes. Ces deux matrices sont définies de la même manière que la matrice R précédemment introduite. Nous appliquons l'analyse et la synthèse en utilisant une matrice résultant du produit tensoriel entre R_l,R_c et une ligne de points, et nous réarrangeons la grille de points sur une seule ligne.

2.3.1 Optimisation

Nous avons mentionné que notre modèle (c'est-a-dire la matrice R) peut être apparenté à un filtre de synthèse utilisé dans la transformée en ondelette, donc nous pouvons l'utiliser pour représenter des objets en multirésolution. Afin d'accomplir cette représentation nous devons optimiser ce modèle avec l'objet que nous voulons représenter (une courbe ou une surface).

Méthode d'optimisation pour les courbes

Dans cette partie, nous considérons qu'une courbe est un ensemble ordonné de points. La méthode d'optimisation est basée sur la minimisation de la distance entre la courbe originale et la courbe reconstruite avec notre modèle en prenant comme paramètres la position des points de contrôle et la matrice R. Nous notons les points de la courbe originale par p_i où $i = 0 \ldots n-1$ et n est le nombre de points initiaux. Nous notons les points générés avec notre modèle par P'_i. Il est important de noter ici que notre modèle permettant de reconstruire parfaitement les données d'entrée, nous allons volontairement

omettre une partie de l'information de détail lors de la reconstruction. La distance entre chaque doublon de points est définie par :

$$d_i = \|p_i - p'_i\| \quad \text{où} \quad i = 0 \dots n-1$$

Nous avons choisi la méthode de Levenberg-Marquardt [PFTV93a] pour minimiser cette distance. Il s'agit d'une méthode de régression non linéaire basée sur la différentiabilité de la fonction à minimiser. Formellement, elle s'exprime de cette manière, soient :

- n points dans le plan $(x_i, y_i)_{i=0\dots n-1}$
- une fonction $y = f(x, a)$, où a représente un vecteur des paramètres.

Le but est de trouver le vecteur de paramètres a_{opt} qui approxime le mieux les données c'est-à-dire :

$$f(x_i, a_{\text{opt}}) \approx (y_i)_{i=0\dots n-1}$$

Dans notre cas, chaque donnée d'entrée (x_i, y_i) va correspondre au couple $(i, 0)$ et la fonction f sera définie par $f(i, a) = d_i$ où a est le vecteur de paramètres représentant dans notre cas la position des points de contrôle et les valeurs de la matrice R.

La fonction de mérite $\mathbb{X}^2(a)$ utilisée dans la méthode de Levenberg-Marquardt est la suivante :

$$\mathbb{X}^2(a) = \sum_{i=0}^{n-1} (0 - f(i, a))^2 = \sum_{i=0}^{n-1} (0 - d_i)^2$$

La méthode étant différentielle, nous devons calculer numériquement la dérivée de f par rapport aux paramètres a_k :

$$\frac{\partial f(i, a)}{\partial a_k} = \frac{f(i, a)_{a_k+\epsilon} - f(i, a)}{\epsilon}$$

avec $k = 0 \dots m-1$ où m est le nombre de paramètres.

Nous avons utilisé la notation $f(i, a)_{a_k+\epsilon}$ pour dire que la valeur a_k du vecteur de paramètres a été modifiée de ϵ. La minimisation s'effectue en deux étapes :

- Dans la première étape, nous minimisons la distance entre la courbe originale et la courbe construite avec notre modèle en utilisant les positions de points de contrôle et les coefficients des matrices de transformation comme paramètres. Dans ce cas, aucune information de détail n'est utilisée pour reconstruire la courbe.
- Dans la deuxième étape, nous minimisons cette distance en prenant les coefficients des matrices de déplacement de détail comme paramètres. Dans ce cas, une partie de l'information de détail seulement est utilisée, sinon la reconstruction serait parfaite et il n'y aurait rien à minimiser. En pratique, les trois derniers niveaux de

détails sont omis.

Les détails utilisés dans l'optimisation sont produits par application de la formule d'analyse $(\mathcal{P}_\theta|\delta\mathcal{P}_\theta) = (\mathcal{P}_{\theta 0}|\ldots|\mathcal{P}_{\theta N-1})\, R^{-1}$ en commençant par les points initiaux jusqu'à arriver aux points de contrôle.

Méthode d'optimisation pour les surfaces

Comme nous avons déjà dit une surface est un ensemble de points $(p_{i,j})_{i,j=0\ldots n-1}$ placés sur une grille où n est la taille de cette grille. Nous avons besoin dans ce cas d'optimiser les deux matrices R_l et R_c. Les étapes d'optimisation sont très proches du cas des courbes. La seule différence est le traitement des matrices R_l et R_c.

2.4 Rsultats

2.4.1 Courbe

Nous avons utilisé ce modèle pour appliquer des déformations sur des courbes. Pour cela nous avons optimisé notre modèle (c'est-a-dire la matrice R) avec la courbe.

La figure 18 montre un exemple de déformation de courbe avec toutes les étapes intermédiaires :

- Optimisation des matrices T_i et U_i au regard de la courbe.
- Analyse de la courbe : obtention d'un polygone de contrôle ainsi que des coefficients de détails.
- Déplacement d'un point de contrôle et calcul de la matrice de déplacement $\mathcal{M}$ associée.
- Application de la partie linéaire de $\mathcal{M}$ $\mathcal{L}(\mathcal{M})$ sur les détails.
- affichage de la courbe déformée.

2.4.2 Surface

Nous optimisons notre modèle avec l'image de Lena (Figure 19-a), puis nous reconstruirons l'image en ignorant 3,2 et 1 niveaux de détails , figures 19-b,19-c et 19-d respectivement.

2.5 Conclusion

Cette approche présente un moyen simple pour la synthèse comme pour l'analyse mais le coût élevé de calcul dans l'étape d'optimisation (surtout si nous travaillons avec les

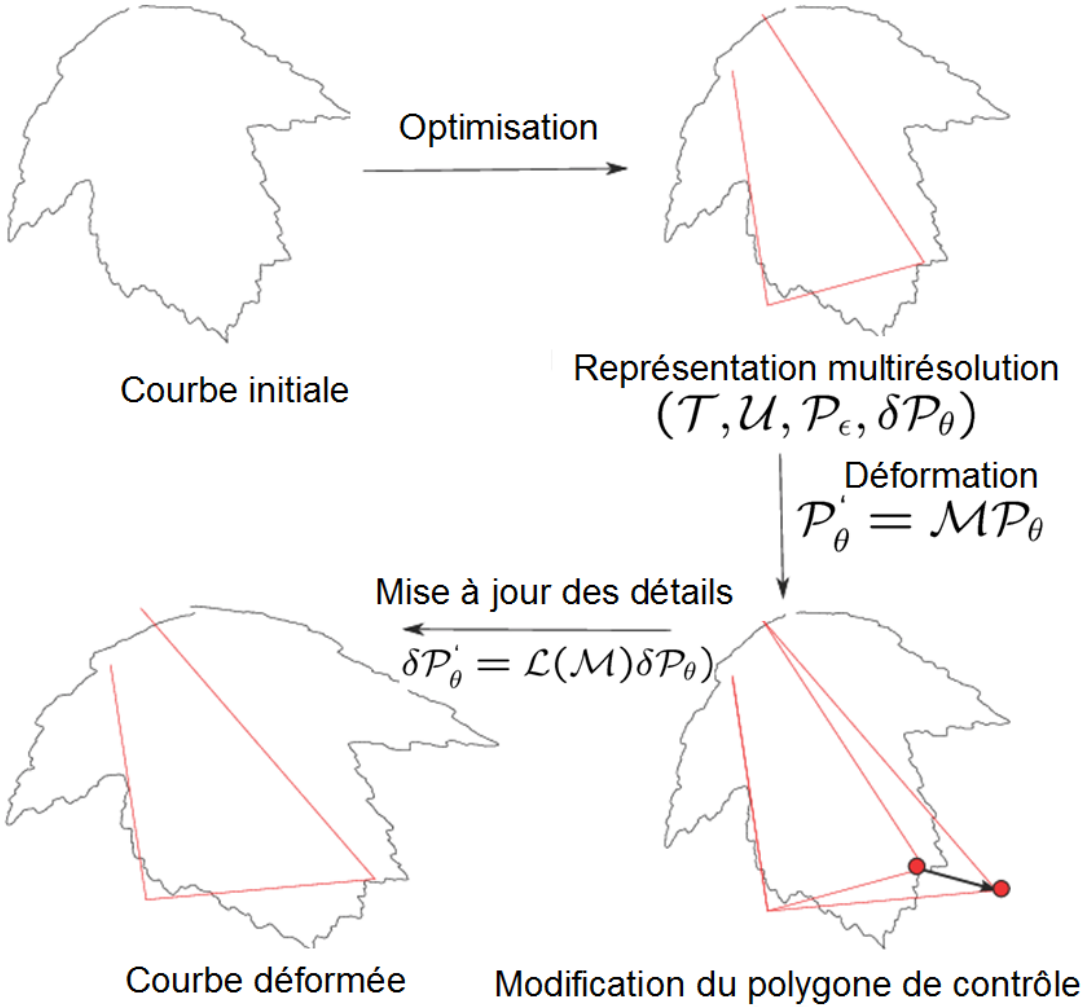


FIG. 18 – Étapes de déformation globale

surfaces) à cause d'une méthode non-linéaire de minimisation, et son incapacité d'effectuer des déformations locales, nous donne la motivation d'introduire une autre approche qui prend en compte ces inconvénients.

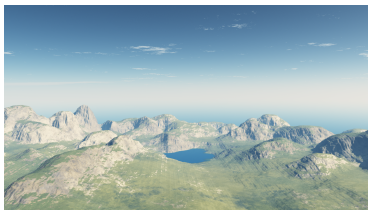
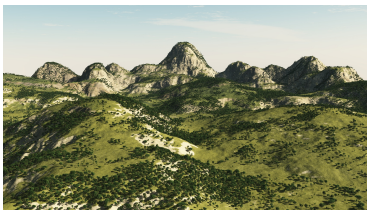
(a) (b)

(c) (d)

FIG. 19 – Reconstruction de l'image de Lena avec notre modèle. (a) Image originale. (b) reconstruction sans les trois derniers niveaux de détails. (c) reconstruction sans les deux derniers niveaux de détails. (d) reconstruction sans le dernier niveau de détails

Chapitre 3

Modèle basé sur la subdivision


Sommaire

L'approche dans cette section introduit la possibilité d'effectuer des déformations locales avec un coût d'optimisation réduit. Cette approche est basée sur le principe de la surface de subdivision [CC78, DS78]. La surface de subdivision est utilisée dans plusieurs champs de recherche (édition de surfaces [KS99], approximation de surfaces [HKD93], *etc*). L'ajout de détails dans les surfaces de subdivision a déjà été introduit dans un but de compression [BDHJ04] ou d'édition multirésolution [FS94, Elb01]. Nous utilisons l'idée de la surface de subdivision avec l'ajout de détails afin de permettre l'édition multirésolution de courbes ou de surfaces. La nouvelle idée dans notre approche est que nous employons plusieurs masques afin d'avoir plus de contrôle sur la forme de la courbe ou de la surface, nos masques peuvent ainsi être des masques fractals.

3.1 Surfaces de subdivision

3.1.1 Principe de la subdivision

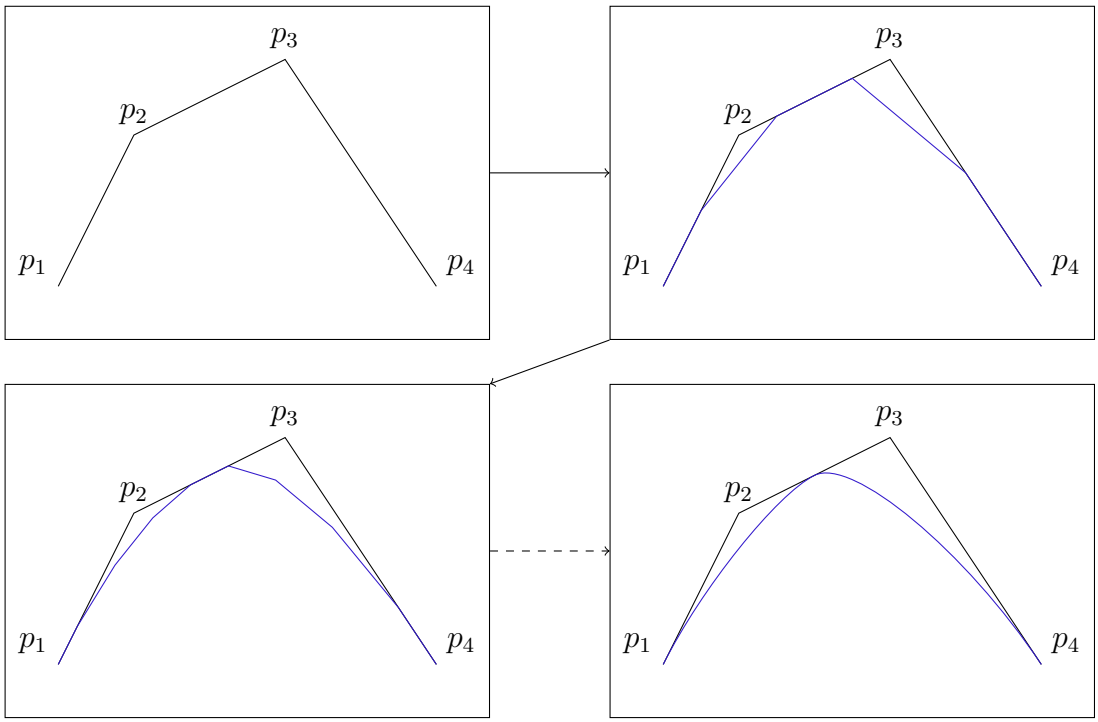


FIG. 20 – Le schéma de subdivision pour générer une courbe comme il est proposé par CHAIKIN [Cha74].

L'idée de subdivision remonte à 1974 quand CHAIKIN [Cha74] a proposé un algorithm pour la génération de courbes arbitraires. Cet algorithme produit simplement une liste de points qui construisent la courbe. La courbe se compose de segments concaténés,

où chacun de ces segments est un ouvert. La courbe peut être arbitrairement complexe, c'est-à-dire qu'elle peut être lisse ou discontinue, et elle peut être ouverte, fermée, ou s'auto-intersecter. Puis en 1978 Catmull et Clark [CC78] ont introduit la technique de surface de subdivision en se basant sur cet algorithme.

Génération de courbe par subdivision

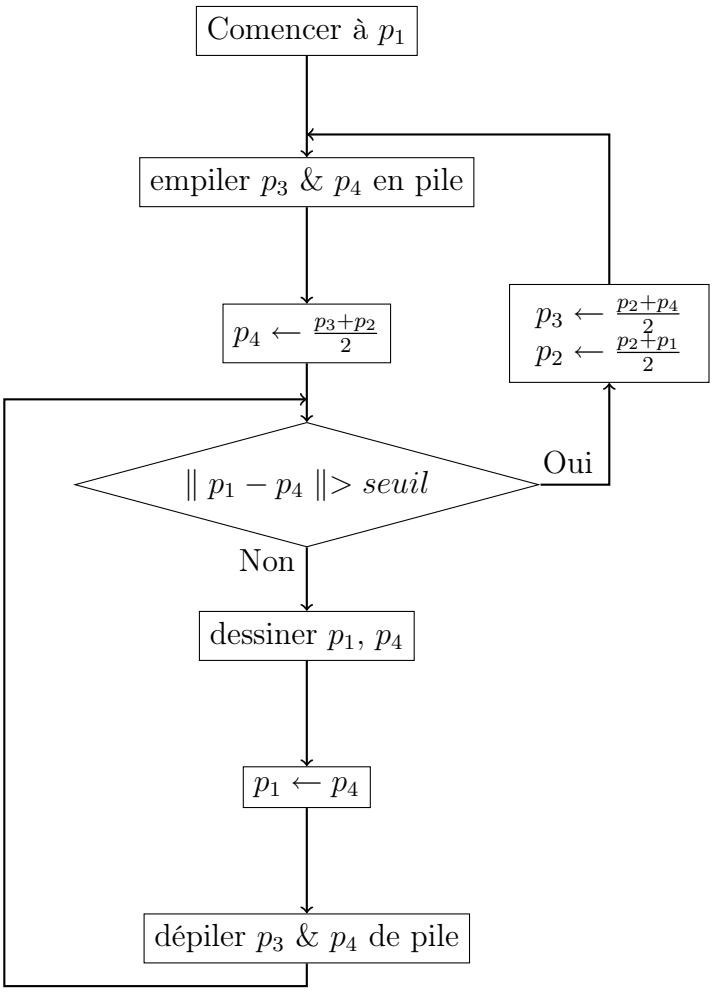


FIG. 21 – L'algorithme de génération de courbe de CHAIKIN.

Nous expliquons la méthode avec le même exemple que celui présenté par CHAIKIN [Cha74]. Considérons une courbe décrite par quatre points (p_1, p_2, p_3, p_4), où les points sont représentés par des paires d'entiers qui spécifient les coordonnées matricielles (raster). La courbe générée part du premier point p_1 en passant par le point milieu des points p_2, p_3 et elle termine au dernier point p_4, les tangentes dans ces trois points sont respectivement $\overrightarrow{p_1p_2}$, $\overrightarrow{p_2p_3}$ et $\overrightarrow{p_3p_4}$ (Figure 21).

Une vue géométrique de l'algorithme considère le quadrilatère décrit par les quatre points et divisé en deux triangles au milieu de la ligne P_2, P_3, chacun de ces triangles est divisé en deux pour former un nouveau quadrilatère. Ce processus se poursuit jusqu'à ce que les deux points du triangle qui sont sur la courbe occupent des points adjacents. La courbe est l'ensemble de tous les vecteurs qui rejoignent ces points. Le schéma de la figure 21 représente les étapes de l'algorithme de la génération d'une courbe tandis que la figure 20 montre les courbes générées pour 1, 2 et n itérations.

Remarque : Même si cet algorithme a été proposé pour traiter des cordonnées en entier, nous pouvons l'utiliser facilement avec des cordonnées en flottant.

Surface de subdivision

En 1978, l'algorithme bien connu de surface de subdivision est introduit par Catmull et Clark [CC78]. L'idée de cet algorithme est la génération de surface de manière récursive. L'idée de base consiste à prendre un morceau de B-Spline cubique défini par un maillage rectangulaire de points de contrôle. Ce morceau est contrôlé par seize points de contrôle (les points noirs dans la figure 22). La subdivision de ce morceau en quatre sous-morceaux génère 25 points de contrôle (les points en couleur dans la figure 22). Les points générés sont divisés en trois catégories :

a) Les points de facette (les points bleus dans la figure 22) : ce sont les points qui se trouvent au milieu de chaque facette du morceau original de B-Spline ;

b) Les points d'arête (les points verts dans la figure 22) : ce sont les points qui se trouvent sur une arête connectant deux points du morceau original ;

c) les points de sommet (les points rouges dans la figure 22) : ce sont les points qui correspondent aux anciens points de contrôle.

En fractionnant le morceau original, nous pouvons noter facilement que chaque nouveau point de contrôle de n'importe quelle type peut être calculé par ses points voisins avec la même expression algébrique. Par exemple un nouveau point de facette est calculé comme la moyenne des quatre anciens sommets qui définissent cette facette.

Nous expliquons cet algorithme premièrement pour le fractionnement d'un morceau de B-Spline, ensuite son extension pour un maillage de topologie arbitraire va être montrée.

Fractionnement d'un morceau de B-Spline

Chaque morceau de B-Spline peut être exprimé sous forme matricielle par :

$$S(u, v) = UMGM^tV^t \tag{3.1}$$

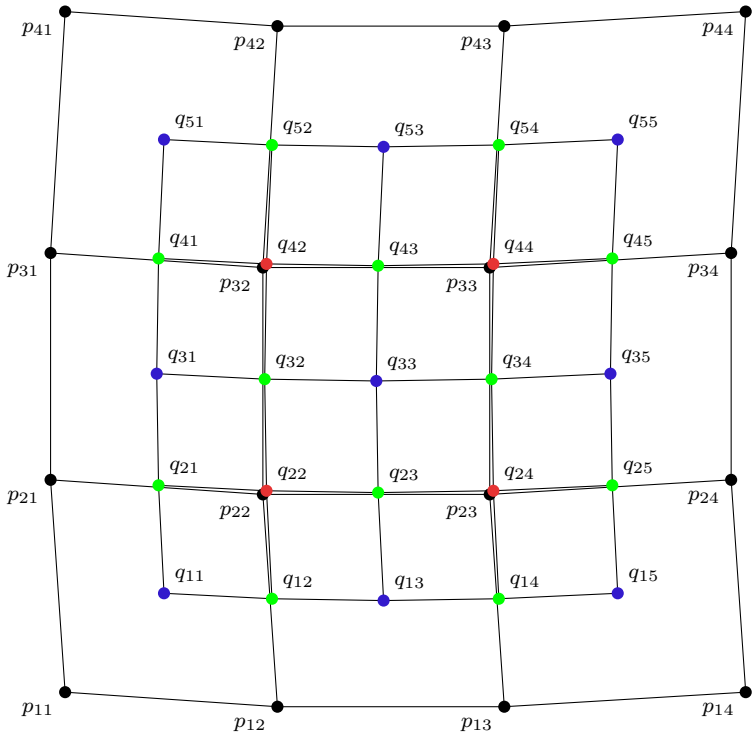


FIG. 22 – Fractionnement d'un morceau de B-Spline défini par un maillage rectangulaire de points de contrôle.

où M est la matrice de base de B-Spline cubique, elle est donnée par :

$$M = \frac{1}{6}\begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix}$$

G est l'ensemble de points de contrôle qui sont arrangés sur un maillage rectangulaire selon leurs indices :

$$G = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{pmatrix}$$

et U et V sont les vecteurs de base :

$$U = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \text{ et } V = \begin{pmatrix} v^3 & v^2 & v & 1 \end{pmatrix}$$

Nous expliquons seulement le calcul concernant le sous-morceau de ce morceau qui correspond au $0 < u, v < \frac{1}{2}$. Les autres sous-morceaux n'ont pas besoin d'être considérés grâce à la symétrie des bases de B-Spline. Le sous-morceau que nous calculons, c'est le morceau $S(u_1, v_1)$ où $u_1 = \frac{u}{2}$ et $v_1 = \frac{v}{2}$. En substituant ces deux expressions dans l'equation (3.1) :

$$S(u_1, v_1) = USMGM^tS^tV^t \tag{3.2}$$

où

$$S = \begin{pmatrix} \frac{1}{8} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

et U et V sont toujours les vecteurs de base :

$$U = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \text{ et } V = \begin{pmatrix} v^3 & v^2 & v & 1 \end{pmatrix}$$

Ce morceau est toujours une B-Spline cubique avec son propre maillage G_1 de points de contrôle satisfaisant :

$$S(u, v) = UMG_1M^tV^t$$

Cela requiert que cette expression doit être égale à l'equation (3.2), ceci sera vrai pour des valeurs arbitraires de u et v si et seulement si :

$$MG_1M^t = SMGM^tS^t$$

Supposons que la matrice de base M soit inversible, ce qui est le cas, nous trouvons que :

$$\begin{aligned} G_1 &= [M^{-1}SM]G[M^tSM^{-t}] \\ &= H_1GH_1^t \end{aligned}$$

où $H_1 = M^{-1}SM$ est appelée la matrice de fractionnement. Apres avoir effectué la multiplication matricielle nous avons :

$$H_1 = \begin{pmatrix} 4 & 4 & 0 & 0 \\ 1 & 6 & 1 & 0 \\ -0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \end{pmatrix}$$

Par conséquent le maillage de points de contrôle correspondant au nouveau sous-morceau est lié à l'ancien maillage de points de contrôle par l'expression :

$$G_1 = H_1 G H_1^t$$

Comme illustré dans la figure 22, les nouveaux points de facette (points bleus) sont calculés par :

$$\left(q_{l,k} = \frac{(p_{i,j} + p_{i,j+1} + p_{i+1,j} + p_{i+1,j+1})}{4} \right)_{\text{où } l,k=\{1,3,5\} \text{ et } i,j=\{1,2,3\}} \tag{3.3}$$

Pareillement, les nouveaux points d'arête sont donnés par :

$$\left(q_{l,k} = \frac{q_{l,k-1} + q_{l,k+1} + p_{i,j} + p_{i+1,j}}{4} \right)_{\text{où } l=\{1,3,5\},\ k=\{2,4\},\ i=\{1,2,3\} \text{ et } j=\{2,3\}} \tag{3.4}$$

et

$$\left(q_{l,k} = \frac{q_{l-1,k} + q_{l+1,k} + p_{i,j} + p_{i,j+1}}{4} \right)_{\text{où } l=\{2,4\},\ k=\{1,3,5\},\ i=\{2,3\} \text{ et } j=\{1,2,3\}} \tag{3.5}$$

Finalement, les nouveaux points de sommet sont donnés par :

$$\left(q_{l,k} = \frac{Q}{4} + \frac{R}{2} + \frac{p_{i,j}}{4} \right)_{\text{où } l,k=\{2,4\} \text{ et } i,j=\{2,3\}} \tag{3.6}$$

où

$$Q = \frac{q_{l-1,k-1} + q_{l-1,k+1} + q_{l+1,k-1} + q_{l+1,k+1}}{4}$$

et

$$R = \frac{1}{4}\left(\frac{p_{i,j} + p_{i,j-1}}{2} + \frac{p_{i,j} + p_{i+1,j}}{2} + \frac{p_{i,j} + p_{i,j+1}}{2} + \frac{p_{i,j} + p_{i-1,j}}{2} \right)$$

Nous pouvons bien noter que tous les nouveaux points sont calculés facilement avec des expressions algébriques en fonction des anciens points.

Fractionnement d'un maillage de topologie arbitraire

Pour que les équations (3.3),(3.4),(3.5) et (3.6) puissent être généralisées pour des maillages de topologie arbitraire, Catmull et Clark [CC78] assument qu'il est plus pratique de les exprimer comme un ensemble de règles qui sont dépendantes du nombre de points autour d'une facette et du nombre d'arêtes incidentes à un sommet . Bien sûr ces règles doivent produire les expressions (3.3),(3.4),(3.5) et (3.6) lorsque ce nombre est égal à quatre.

Les règles de fractionnement deviennent ainsi :

- Les nouveaux points de facette : la moyenne de tous les anciens points définissant

la facette.

- Les nouveaux points d'arête : la moyenne du milieu de l'ancienne arête avec la moyenne des deux nouveaux points de facette des facettes qui partagent l'arête.
- Les nouveaux points de sommet : la moyenne

$$\frac{Q}{n} + \frac{2R}{n} + \frac{S(n-3)}{n}$$

où

- Q = la moyenne des nouveaux points de facette de toutes les facettes adjacentes des anciens points de sommet.
- R = la moyenne des milieux de toutes les anciennes arêtes incidentes dans l'ancien point de sommet.
- S = l'ancien point de sommet.

Après le calcul de ces points, de nouvelles arêtes sont formées par :

- la connexion de chacun des nouveaux points de facette aux nouveaux points d'arête des arêtes définissant l'ancienne facette.
- la connexion de chacun des nouveaux points de sommet aux nouveaux points d'arête de toutes les arêtes incidentes dans l'ancien point de sommet.

Des nouvelles facettes seront définies comme celles qui sont enfermées par les nouvelles arêtes.

La figure 23 montre un exemple de l'utilisation de cet algorithme afin de générer une surface lisse qui approxime les sommets de départ.

Un autre méthode pour la subdivision a été présentée par Doo-Sabin [DS78]. Cette méthode a été proposée à la même époque que la méthode de Catmull-Clark, et elle est une avancée très importante dans l'histoire de la surface de subdivision. Elle traite le comportement des points extraordinaires qui sont les points partagés par n arêtes avec $n \neq 4$. Depuis plusieurs techniques ont été proposées en se basant sur la subdivision afin de générer une surface lisse qui approxime les sommets de départ. La figure 24 est un exemple de la subdivision de Doo-Sabin, notons que les points extraordinaires sont traités différemment.

Les méthodes de Catmull-Clark et Doo-Sabin traitent des maillages rectangulaires. Loop [Loo87] a introduit une technique de subdivision pour les surfaces représentées par un maillage triangulaire. La figure 25 montre comment cette technique est utilisée pour raffiner un icosaèdre.

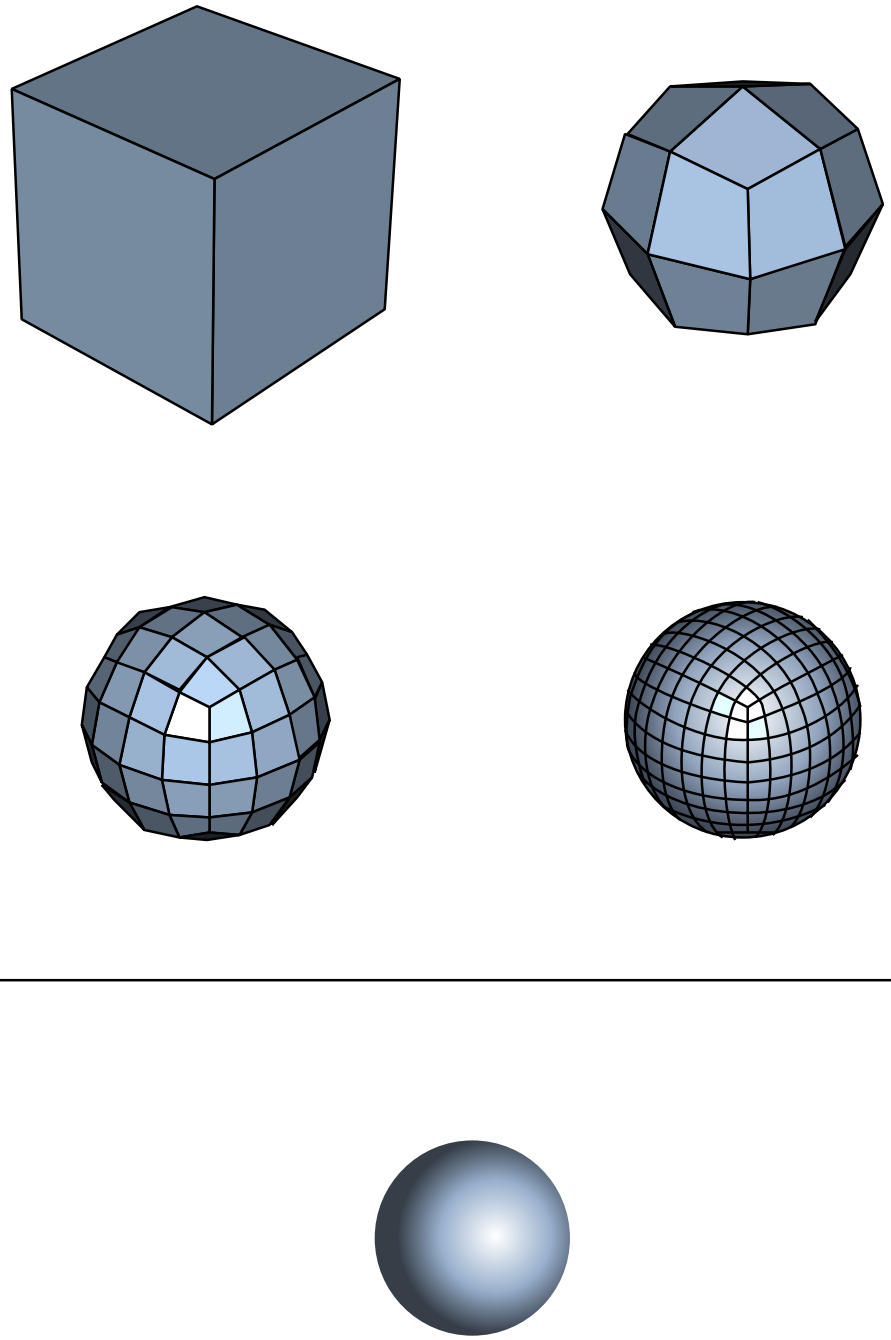


FIG. 23 – Les trois premières étapes de subdivision de Catmull-Clark d'un cube et la surface finale : une sphère.

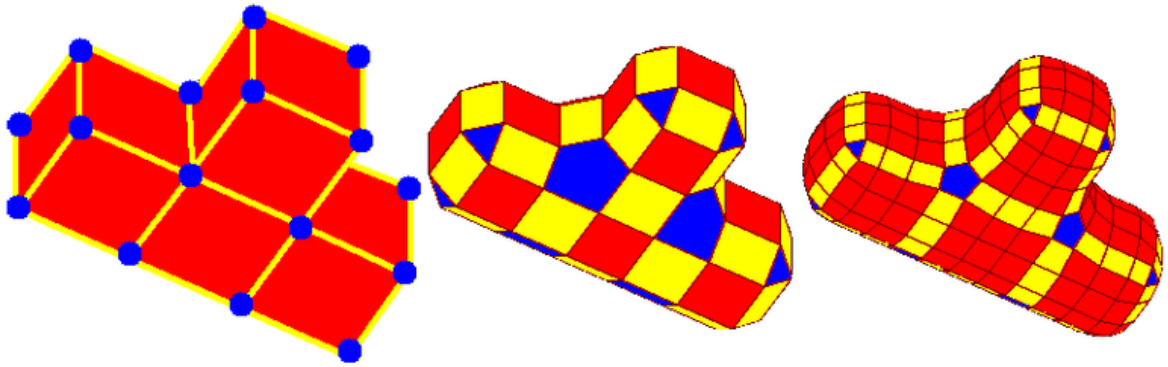


FIG. 24 – Une surface de subdivision de Doo-Sabin.

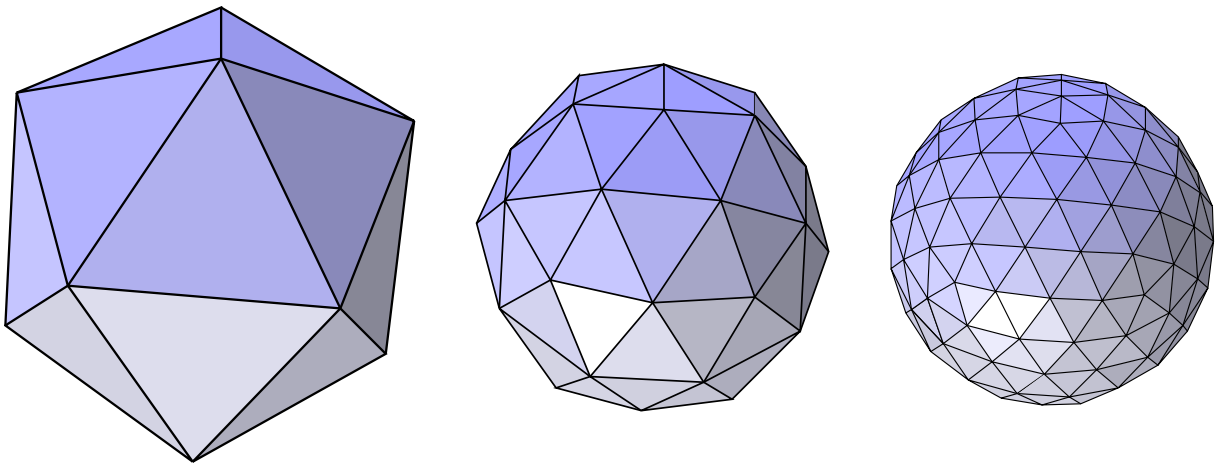


FIG. 25 – Loop, subdivision d'un icosaèdre (à gauche) après une et après deux étapes de raffinement.

3.1.2 Exemple

La figure 26 montre un exemple de l'utilisation de l'algorithme de subdivision pour construire une surface lisse.

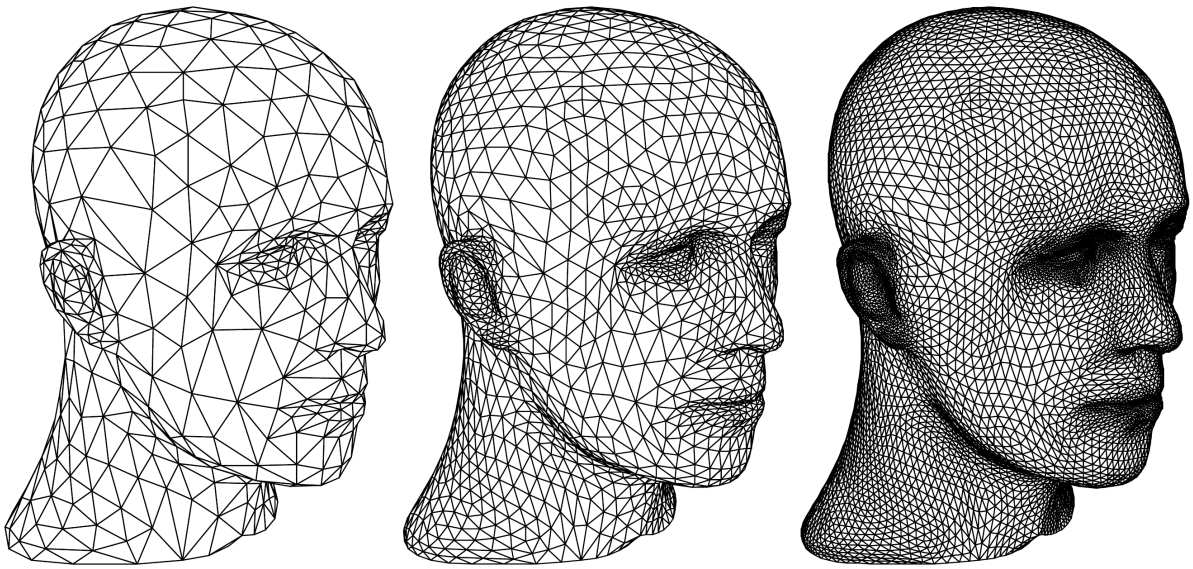


FIG. 26 – Un exemple de la subdivision de surface.

3.1.3 Conclusion

Dans cette section nous avons présenté l'idée de base de subdivision, nous avons détaillé l'algorithme de subdivision introduit par Catmull-Clark parce que cet algorithme est la base de notre méthode de subdivision. Notre contribution pour la subdivision est l'utilisation de masques de subdivision qui sont des masques lisses ou fractals. Dans la section suivante nous allons expliquer le principe de subdivision utilisé dans notre méthode.

3.2 Notre modèle de subdivision

Dans cette section nous expliquons notre modèle de subdivision. Nous commençons par le cas des courbes, puis nous considérons le cas des surfaces définies par une grille de points de contrôle. Finalement nous expliquons comment nous pouvons généraliser ce modèle pour subdiviser un maillage de topologie arbitraire.

3.2.1 Modèle de subdivision de courbes

Notre modèle de subdivision de courbe, comme tous les autres modèles de subdivision, génère une courbe en subdivisant le polygone de points de contrôles de départ successivement. La courbe générée passe par tous les points de contrôles sauf le premier et le dernier point.

Soit $(p_i)_{i=0,..,m-1}$ un ensemble de points de contrôle, notre opérateur de subdivision est défini par : $M = ((\frac{1}{2}\ \frac{1}{2}), (1), (a\ b\ c\ d))$ tel que $a + b + c + d = 1$. Nous notons que l'opérateur se compose de trois types de masques, chacun de ces masques est utilisé pour calculer une partie des points générés. Nous notons $(q_j)_{j=0,..,n-1}$ les points générés, que nous calculons comme ceci :

$$q_j = \begin{cases} (\frac{1}{2}\ \frac{1}{2})(p_i\ p_{i+1})^t & \text{si } i = j = 0 \text{ ou } i = m-2 \text{ et } j = 2i = n-1 \\ p_i & \text{si } 0 < i < m-1 \text{ et } j = 2i-1 \\ (a\ b\ c\ d)(p_i\ p_{i+1}\ p_{i+2}\ p_{i+3})^t & \text{si } i < m-3 \text{ et } j = 2i+2 \end{cases} \tag{3.7}$$

Nous notons que notre opérateur de subdivision possède un masque ayant quatre coefficients, ainsi notre modèle doit commencer avec au minimum quatre points de contrôle. Chaque étape de subdivision génère $n = 2m-3$ points où m est le nombre de points dans l'étape précédente.

L'avantage de notre modèle de subdivision est que nous pouvons jouer avec les coefficients a, b, c, d pour changer le comportement de subdivision. La figure 27 montre une courbe fractale générée en utilisant notre modèle avec $M = ((\frac{1}{2}\ \frac{1}{2}), (1), (0.2\ 0.3\ 0.3\ 0.2))$, alors que la figure 28 qui emploie le même polygone de contrôle que celui de la figure 27, montre une courbe lisse générée avec $M = ((\frac{1}{2}\ \frac{1}{2}), (1), (-0.05\ 0.55\ 0.55\ -0.05))$.

Exemples

La figure 29 montre des exemples de courbes fractales générées avec notre modèle. Chacune de ces courbes utilise un polygone de contrôle et un opérateur de subdivision différents des autres courbes. La figure 30 montre des exemples de courbes lisses générées avec notre modèle. Chacune de ces courbes utilise un polygone de contrôle et un opérateur de subdivision différents des autres courbes.

3.2.2 Modèle de subdivision de surfaces

Dans cette section nous développons notre modèle de subdivision de courbes pour traiter le cas de surfaces. Premièrement nous expliquons la méthode de subdivision en

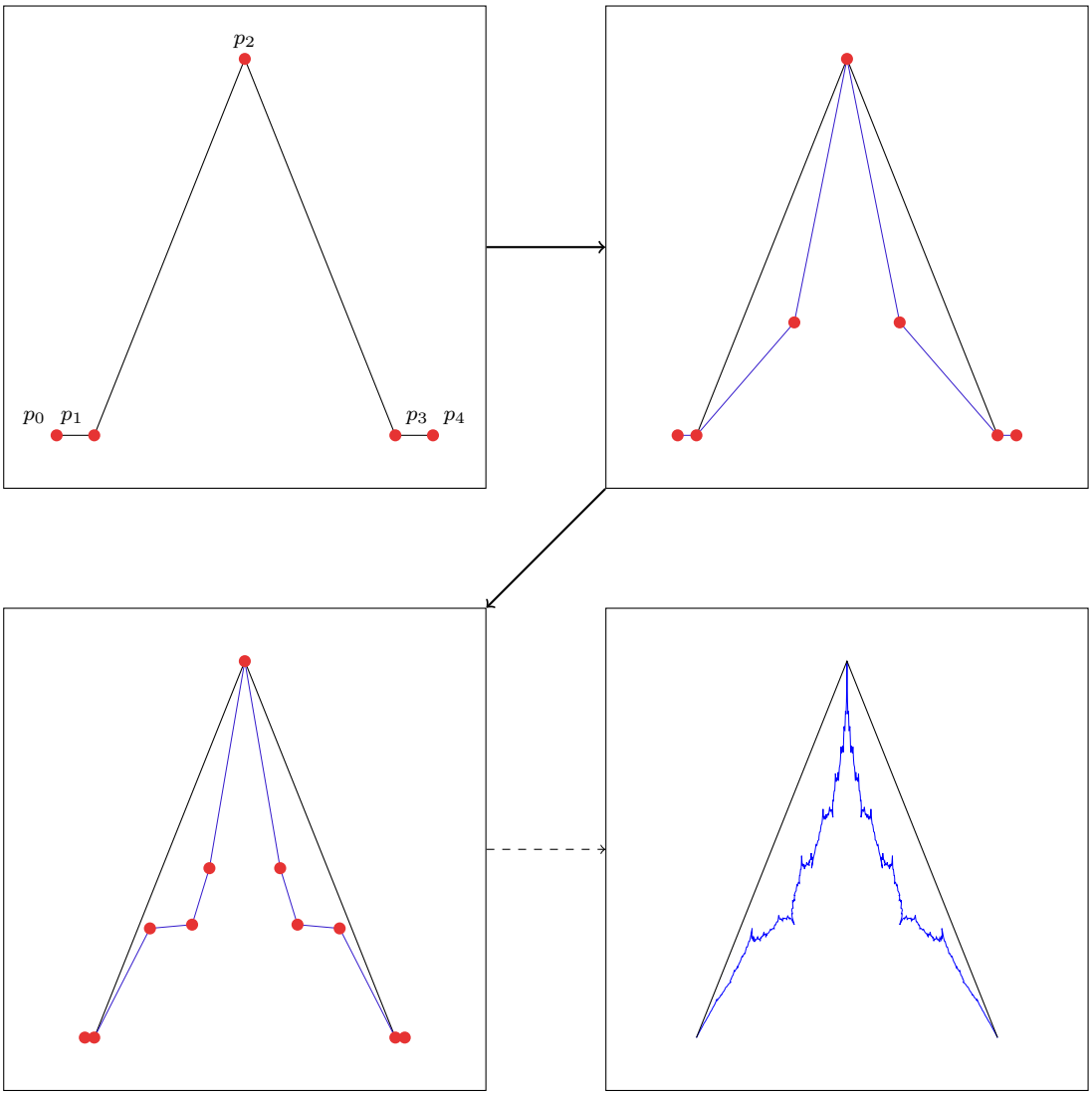


FIG. 27 – Notre modèle de subdivision (courbe fractale).

considérant que la surface est définie par une grille de points de contrôle. Puis nous généralisons cette méthode pour traiter le cas des surfaces de topologie arbitraire.

Surface définie par une grille de points de contrôle

Soit $(p_{i,j})_{(i,j=0,\ldots,m-1)}$ une grille de points de contrôle, une nouvelle grille de points $(q_{k,l})_{(l,k=0,\ldots,n-1)}$ où $n = 2m - 3$ peut être générée en subdivisant la grille $p_{i,j}$. La figure 31 montre une étape de subdivision d'une grille de 16 points de contrôle en une grille de 25 points de contrôle. La nouvelle grille est générée en utilisant un opérateur de subdivision ayant plusieurs masques. Nous avons colorié différemment les points dans la figure 31 selon les masques utilisés pour générer ces points. Nous générons un nouveau point en

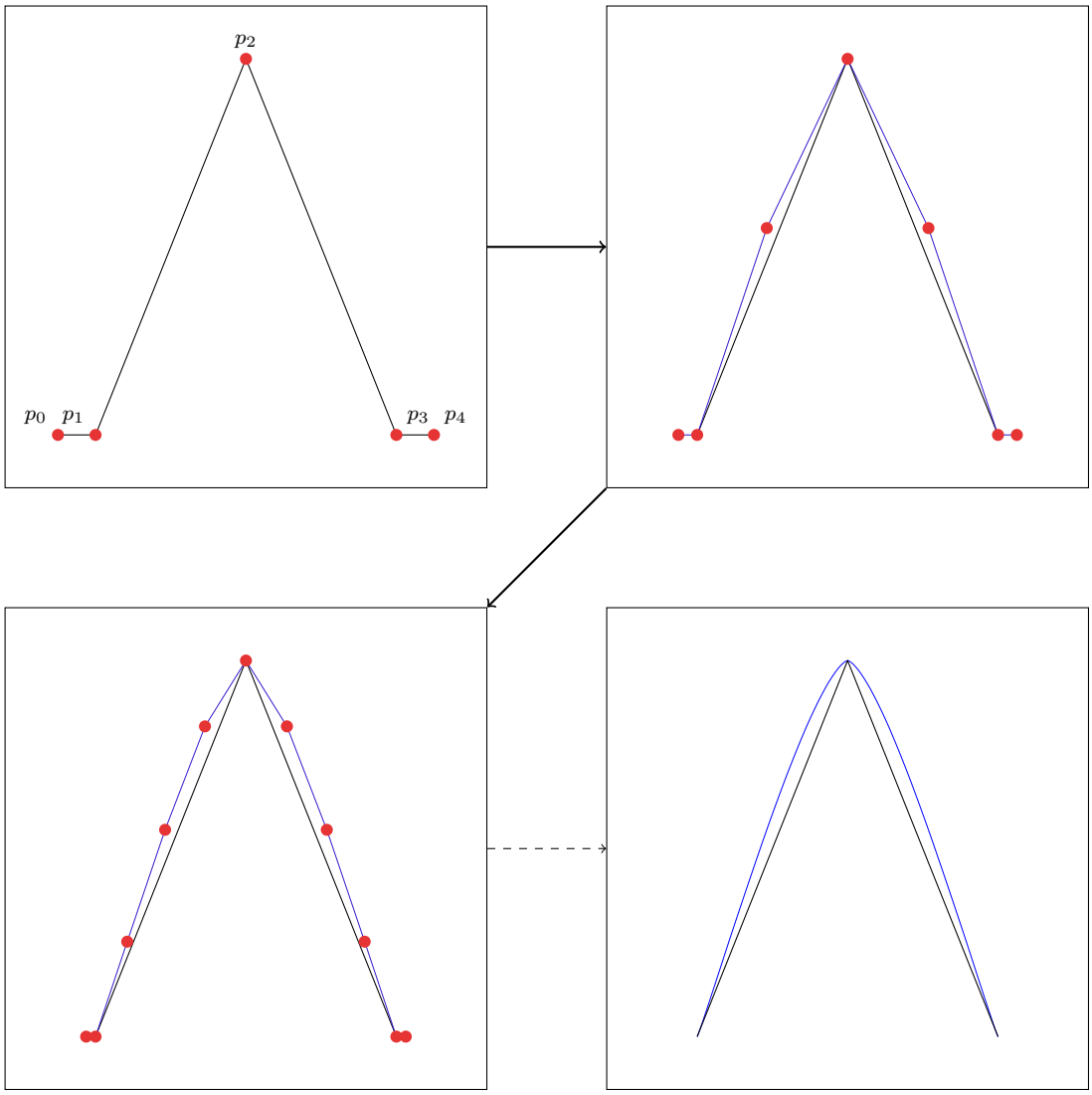


FIG. 28 – Notre modèle de subdivision (courbe lisse).

utilisant la formule suivante :

$$q_{l,k} = \mathcal{M} * P \tag{3.8}$$

où $\mathcal{M}$ est notre opérateur de subdivision, et P est une matrice de points qui sont choisis parmi les points de la grille $(p_{i,j})_{(i,j=0,\ldots,m-1)}$. La dimension de $\mathcal{M}$ est la même que P. Le produit (*) entre $\mathcal{M}$ et P est défini par :

$$\mathcal{M} * P = \sum_{i,j=0}^{L,C} \mathcal{M}_{i,j} P_{i,j}$$

où L et C sont le nombre de lignes et colonnes respectivement de masques $\mathcal{M}$. $\mathcal{M}$, P sont définis comme ceci :

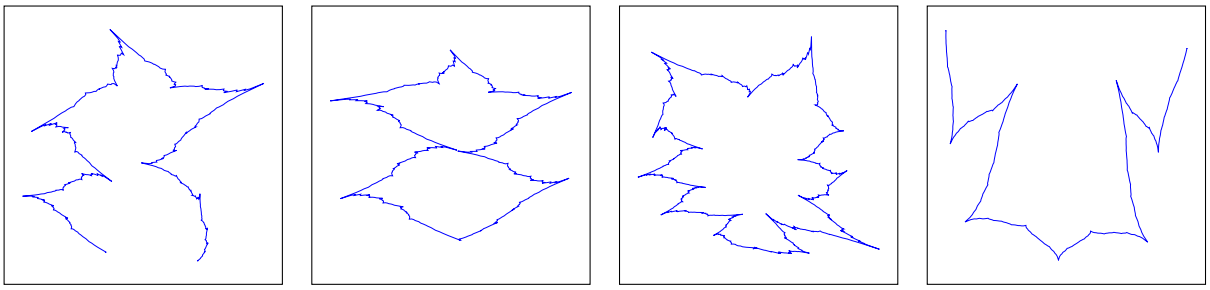


FIG. 29 – Exemple de courbes fractales générées avec notre modèle de subdivision.

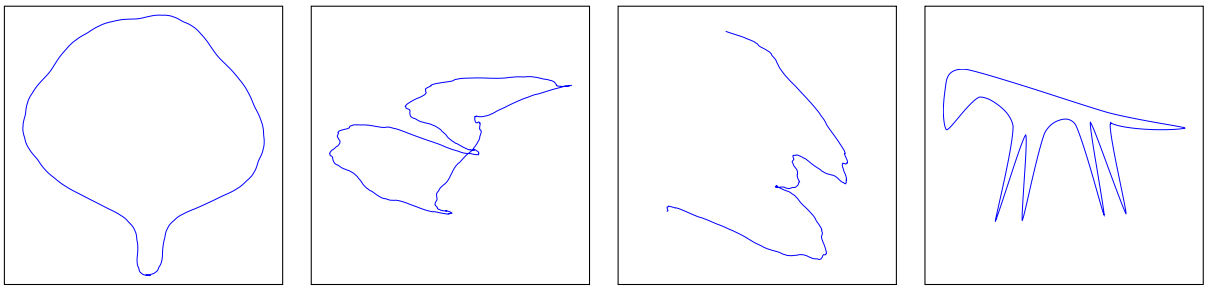


FIG. 30 – Exemple de courbes lisses générées avec notre modèle de subdivision.

- Les quatre points de coins (les points oranges dans la Figure 31) de $q_{k,l}$ sont calculés en utilisant un masque $\mathcal{M} = \mathcal{M}_{22}$ et :

$$P = \begin{pmatrix} p_{i,j} & p_{i,j+1} \\ p_{i+1,j} & p_{i+1,j+1} \end{pmatrix}$$

  où $i = \{0, m-2\}$, $j = \{0, n-2\}$, $l = 2i$ et $l = 2i, k = 2j$;
- Tous les points de bord vertical avec des indices impairs de lignes (les points gris dans la Figure 31) sont calculés en utilisant un masque $\mathcal{M} = \mathcal{M}_{12}$ et :

$$P = \begin{pmatrix} p_{i,j} & p_{i,j+1} \end{pmatrix}$$

  où $j = \{0, n-2\}, 0 < i < m-1$ et $l = 2i-1, k = 2j$;
- Tous les points de bord horizontal avec des indices impairs de colonnes (les points gris dans la Figure 31) sont calculés de manière similaire avec $\mathcal{M} = \mathcal{M}_{21}$ et :

$$P = \begin{pmatrix} p_{i,j} \\ p_{i+1,j} \end{pmatrix}$$

  où $i = \{0, m-2\}, 0 < j < n-1$ et $k = 2j-1, l = 2i$;
- Tous les points de bord vertical avec des indices pairs de lignes (les points verts

dans la Figure 31) sont calculés en utilisant un masque $\mathcal{M} = \mathcal{M}_{42}$ et :

$$P = \begin{pmatrix} p_{i,j} & p_{i,j+1} \\ \vdots & \vdots \\ p_{i+3,j} & p_{i+3,j+1} \end{pmatrix}$$

où $j = \{0, n-2\}, i < m-3$ et $l = 2i+2, k = 2j$;

- Tous les points de bord horizontal avec des indices pairs de colonnes (les points verts dans la Figure 31) sont calculés de manière similaire avec $\mathcal{M} = \mathcal{M}_{24}$ et :

$$P = \begin{pmatrix} p_{i,j} & \cdots & p_{i,j+3} \\ p_{i+1,j} & \cdots & p_{i+1,j+3} \end{pmatrix}$$

où $i = \{0, m-2\}, j < m-3$ et $k = 2j+2, l = 2i$;

- Tous les points d'indices impairs de lignes avec des indices impairs de colonnes (les points noirs dans la Figure 31) sont copiés de $p_{i,j}$ en utilisant un masque $\mathcal{M} = \mathcal{M}_{11}$ et :

$$P = (p_{i,j})$$

où $0 < i < m-1$, $0 < j < n-1$ et $l = 2i-1, k = 2j-1$;

- Tous les points d'indices impairs de lignes avec des indices pairs de colonnes (les points bleus dans la Figure 31) sont calculés en utilisant un masque $\mathcal{M} = \mathcal{M}_{34}$ et :

$$P = \begin{pmatrix} p_{i-1,j} & \cdots & p_{i-1,j+3} \\ p_{i,j} & \cdots & p_{i,j+3} \\ p_{i+1,j} & \cdots & p_{i+1,j+3} \end{pmatrix}$$

où $0 < i < m-1, j < n-3$ et $l = 2i-1, k = 2j+2$;

- Tous les points d'indices pairs de lignes avec des indices impairs de colonnes (les points bleus dans la Figure 31) sont calculés de manière similaire avec $\mathcal{M} = \mathcal{M}_{43}$ et :

$$P = \begin{pmatrix} p_{i,j-1} & p_{i,j} & p_{i,j+1} \\ \vdots & \vdots & \vdots \\ p_{i+3,j-1} & p_{i+3,j} & p_{i+3,j+1} \end{pmatrix}$$

où $0 < j < n-1, i < m-3$ et $l = 2i+2, k = 2j-1$;

- Finalement tous les points d'indices pairs de lignes avec des indices pairs de colonnes (les points rouges dans la Figure 31) sont calculés en utilisant un masque

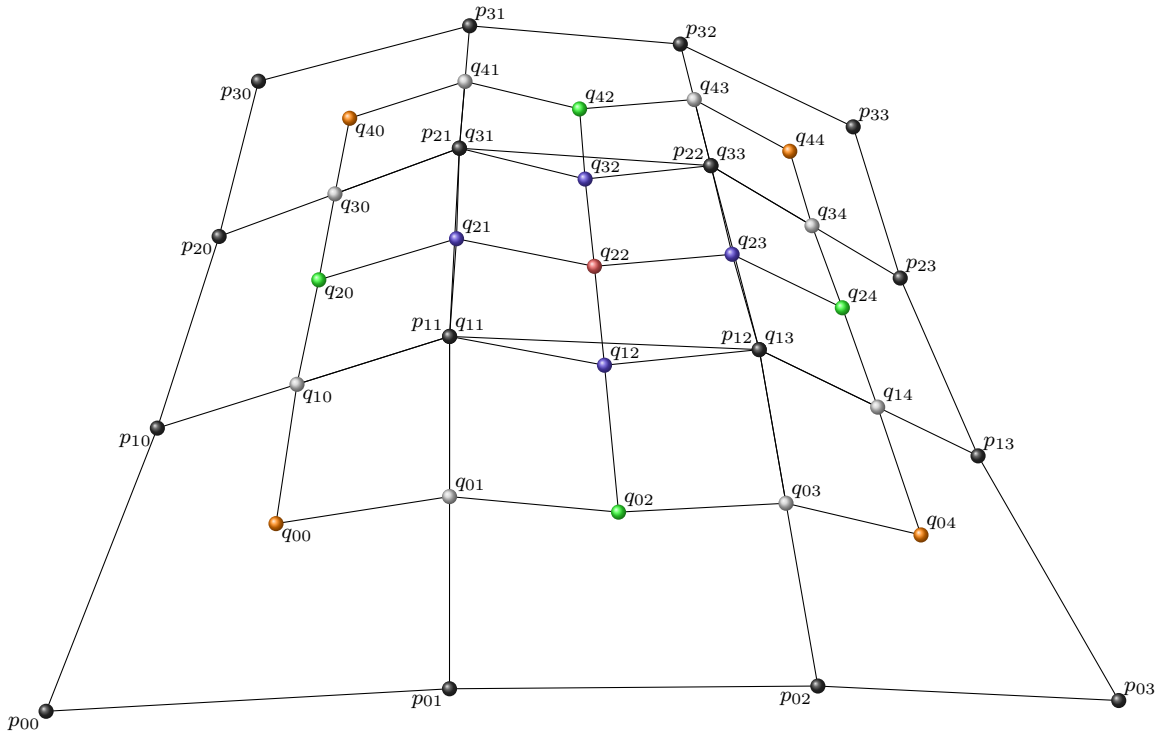


FIG. 31 – Notre modèle de subdivision (surface fractale).

$\mathcal{M} = \mathcal{M}_{44}$ et :

$$P = \begin{pmatrix} p_{i,j} & \cdots & p_{i,j+3} \\ \vdots & \ddots & \vdots \\ p_{i+3,j} & \cdots & p_{i+3,j+3} \end{pmatrix}$$

où $i < m-3, j < n-3$ et $l = 2i+2, k = 2j+2$.

Remarque : Nous pouvons générer une surface en appliquant notre opérateur de subdivision de courbe, il suffit d'appliquer l'opérateur premièrement par lignes, puis par colonnes ou inversement. Mais l'avantage de notre opérateur de surface est que nous pouvons traiter les lignes et les colonnes différemment en utilisant de masques différents.

Exemples

Figure 32 montre des exemples de surfaces fractales générées avec notre modèle. Chacune de ces surfaces utilise une grille de contrôle et un opérateur de subdivision différents des autres surfaces.

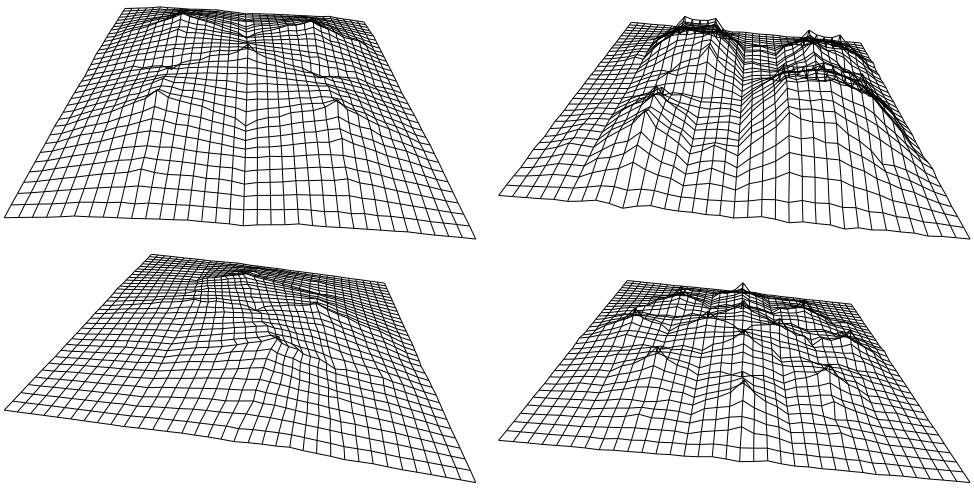


FIG. 32 – Exemple de surfaces fractales générés avec notre modèle de subdivision.

Surface définie par un maillage de topologie arbitraire

Nous supposons que le maillage est un maillage fermé pour expliquer notre modèle, mais nous pouvons sans problème utiliser ce modèle avec un maillage non fermé en traitant le bord de maillage comme un cas spécial. Nous présentons premièrement le cas d'un maillage quadrangulaire, ensuite nous expliquons le cas d'un maillage triangulaire.

Maillage quadrangulaire

Afin de traiter ce genre de surface, nous devons construire la topologie de son maillage de manière à ce que chaque facette soit accompagnée de ses quatre facettes voisines en plus des quatre sommets qui la définissent, et chaque arête soit accompagnée de ses deux facettes voisines en plus des deux sommets qui la définissent.

Nous considérons que deux facettes sont voisines si elles partagent une arête et qu'une facette est voisine à une arête si cette facette contient cette arête.

> **Remarque :** Nous pouvons facilement traiter le maillage non fermé en considérant que chaque facette de bord ne possède que les facettes voisines qui sont dans le maillage, et que chaque arête de bord ne possède qu'une facette voisine. Par exemple toutes les facettes de coin ont seulement deux facettes voisines.

D'une façon similaire à Catmull-Clarck [CC78], chaque étape de subdivision se compose de deux passes sur le maillage, la première passe parcourt toutes les facettes et pour chacune de ces facettes, elle construit un nouveau sommet que nous appelons le sommet de facette. La deuxième passe parcourt toutes les arêtes, et un sommet appelé le sommet d'arête est construit pour chaque arête. Douze sommets sont utilisés afin de calculer chacun des

sommets des facettes, tandis que chaque sommet d'arête a besoin de six sommets pour être calculé. La figure 33 montre les sommets utilisés pour calculer un nouveau sommet. Les sommets qui sont coloriés en magenta influencent le nouveau sommet plus que les autres sommets (les sommets verts).

Supposons que $(p_i)_{(i=0,\ldots,n-1)}$ sont les sommets de maillage avant la subdivision et ce maillage se compose de f facettes et de e arêtes. Alors le maillage après la subdivision possède $m = n + f + e$ sommets. Chaque sommet de facette $q_j(f)$ est calculé comme ceci :

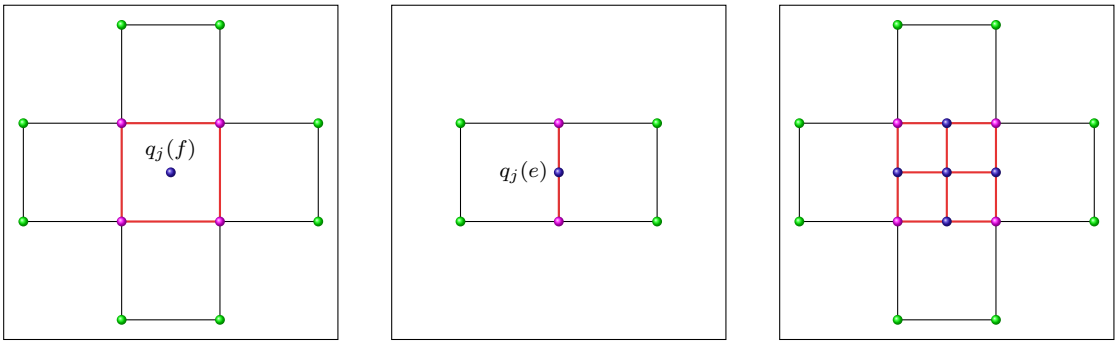
$$q_j(f) = \begin{pmatrix} 0 & a_5 & a_6 & 0 \\ a_3 & a_1 & a_2 & a_8 \\ a_9 & a_3 & a_4 & a_{10} \\ 0 & a_{11} & a_{12} & 0 \end{pmatrix} * \begin{pmatrix} 0 & \bullet & \bullet & 0 \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet & 0 \end{pmatrix}$$

où f est la facette définie par les quatre sommets coloriés en magenta comme dans la figure 33-a et $\sum_{i=1}^{12} a_i = 1$. Chaque sommet d'arête $q_j(e)$ est calculé comme ceci :

$$q_j(e) = \begin{pmatrix} b_3 & b_1 & b_4 \\ b_5 & b_2 & b_6 \end{pmatrix} * \begin{pmatrix} \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{pmatrix}$$

où e est l'arête définie par les deux sommets coloriés en magenta comme dans la figure 33-b et $\sum_{i=1}^{6} b_i = 1$.

Après avoir calculé les nouveaux sommets, la topologie est construite telle que chaque facette du maillage précédent se divise en quatre facettes, le sommet de facette sera un sommet partagé par les quatre nouvelles facettes (Figure 33-c).



(a) (b) (c)

FIG. 33 – Le calcul d'une nouveau sommet (maillage quadrangulaire). (a) Sommet de facette. (b) Sommet d'arête. (c) résultat de subdivision.

Exemples

Dans la figure 34, trois étapes de subdivision sont appliquées sur un cube représenté par un maillage quadrangulaire, les masques utilisés sont des masques qui produisent une influence fractale très apparente. En revanche dans la figure 35, le même maillage est divisé en employant des masques produisant des effets quasiment lisses. La figure 36 montre un autre maillage généré avec notre modèle de subdivision.

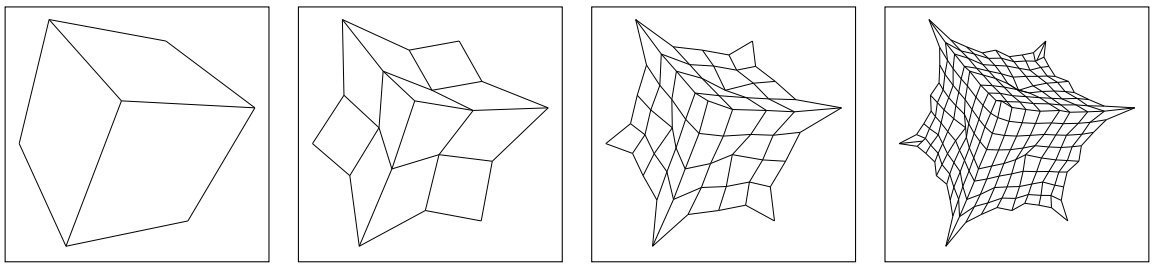


FIG. 34 – Exemple de subdivision de maillage quadrangulaire généré avec notre modèle de subdivision.

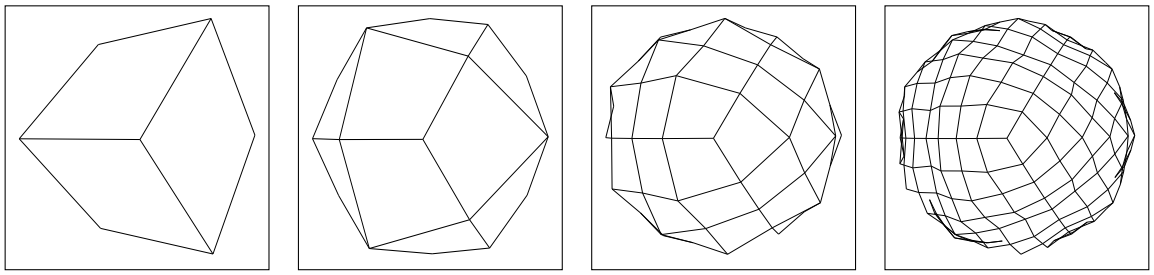


FIG. 35 – Autre exemple de subdivision de maillage quadrangulaire généré avec notre modèle de subdivision. Les masques utilisés sont moins fractals

Maillage triangulaire

Comme pour la surface définie par un maillage quadrangulaire, nous devons construire la topologie du maillage triangulaire de manière à ce que chaque triangle soit accompagné de ses trois triangles voisins en plus des trois sommets qui le définissent, et chaque arête soit accompagnée de ses deux triangles voisins en plus des deux sommets qui la définissent.

De manière similaire au cas de subdivision de maillage quadrangulaire, chaque étape de subdivision se compose de deux passes sur le maillage, la première passe parcourt tous les triangles et pour chacun de ces triangles, elle construit un nouveau sommet que nous appelons le sommet de facette. La deuxième passe parcourt toutes les arêtes, et un sommet appelé le sommet d'arête est construit pour chaque arête. Six sommets sont utilisés afin de calculer chacun des sommets des facettes, tandis que chaque sommet d'arête a besoin de

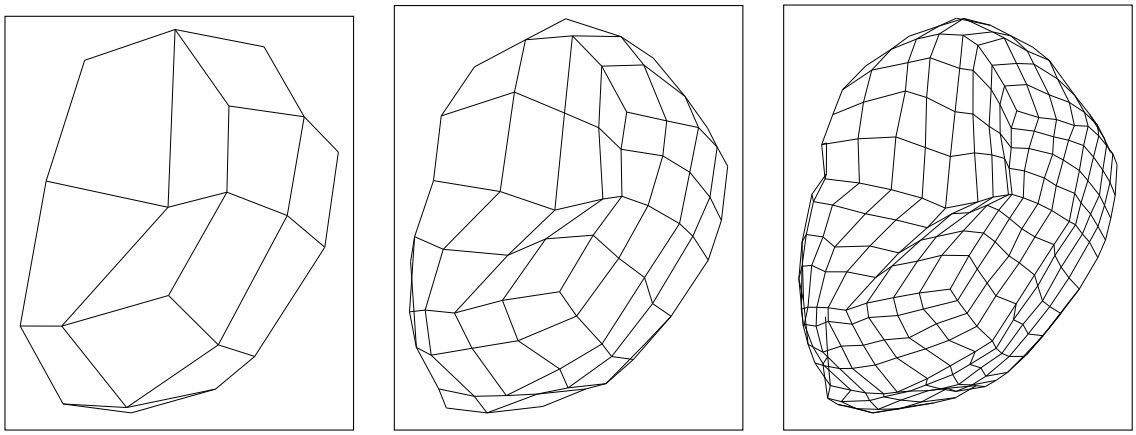


FIG. 36 – Subdivision de maillage quadrangulaire généré avec notre modèle de subdivision.

quatre sommets pour être calculé. La figure 37 montre les sommets utilisés pour calculer un nouveau sommet. Les sommets qui sont coloriés en magenta influencent le nouveau sommet plus que les autres sommets (les sommets verts).

Soit $(p_i)_{(i=0,\ldots,n-1)}$ les sommets de maillage avant la subdivision. Ce maillage se compose de f triangles et de e arêtes. Alors le maillage après la subdivision possède $m = n + f + e$ sommets. Chaque sommet de facette $q_j(f)$ est calculé comme ceci :

$$q_j(f) = \begin{pmatrix} 0 & 0 & a_4 \\ 0 & a_1 & a_2 \\ a_5 & a_3 & a_6 \end{pmatrix} * \begin{pmatrix} 0 & 0 & \bullet \\ 0 & \bullet & \bullet \\ \bullet & \bullet & \bullet \end{pmatrix}$$

où f est le triangle défini par les trois sommets coloriés en magenta comme dans la figure 37-a et $\sum_{i=1}^{6} a_i = 1$. Chaque sommet d'arête $q_j(e)$ est calculé comme ceci :

$$q_j(e) = \begin{pmatrix} 0 & b_1 & b_3 \\ b_4 & b_2 & 0 \end{pmatrix} * \begin{pmatrix} 0 & \bullet & \bullet \\ \bullet & \bullet & 0 \end{pmatrix}$$

où e est l'arête définie par les deux sommets coloriés en magenta comme dans la figure 37-b et $\sum_{i=1}^{4} b_i = 1$.

Après avoir calculé les nouveaux sommets, la topologie est construite telle que chaque triangle du maillage précédent se divise en six triangles, le sommet de facette sera un sommet partagé par les six nouveaux triangles.

Exemples

Dans les figures 38 et 39, deux étapes de subdivision sont appliquées sur un tétraèdre et

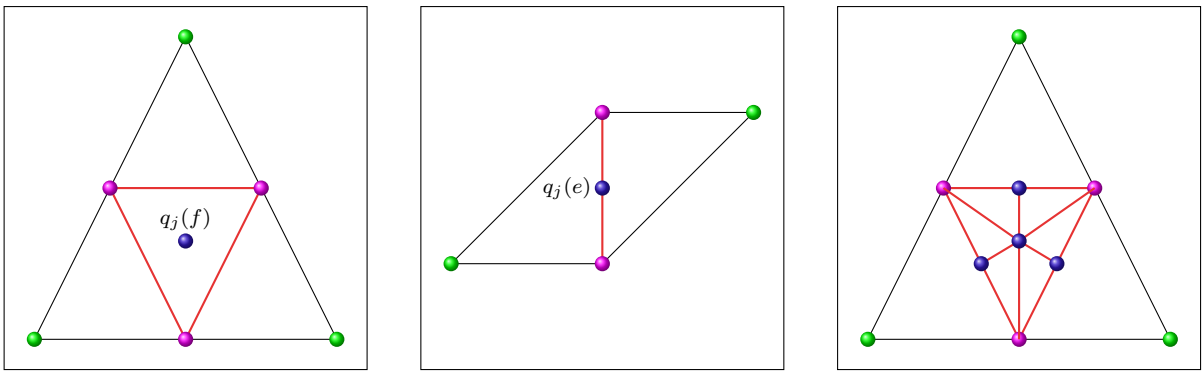


FIG. 37 – Le calcul d'une nouveau sommet (maillage triangulaire). (a) Sommet de facette. (b) Sommet d'arête. (c) résultat de subdivision.

un cube représentés par des maillages triangulaires, les masques utilisés sont des masques qui produisent une influence fractale très apparente pour le cube tandis que la subdivision de tétraèdre emploie des masques qui produisent des effets moins fractals.

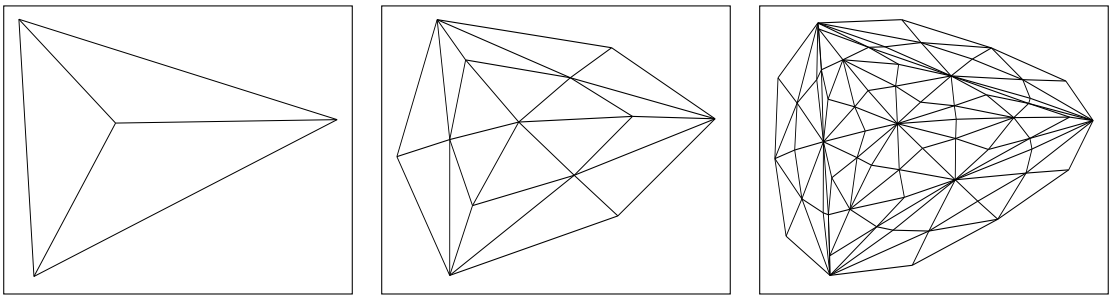


FIG. 38 – Exemple de subdivision de maillage triangulaire générés avec notre modèle de subdivision.

3.3 Modèle de subdivision avec notion de détail

Dans la section précédente nous avons introduit notre modèle de subdivision pour les courbes et aussi pour les surfaces. Dans cette section nous développons ce modèle de subdivision en l'étendant avec une notion de détail. Ce modèle présente un grand avantage pour le contrôle de forme des objets, surtout pour la modification locale pour les courbes et surfaces. L'idée de multirésolution est déjà utilisée par plusieurs méthodes comme la courbe multirésolution [FS94, Elb01] et la subdivision multirésolution [BMBZ02, FB88, MM01], *etc*, mais toujours pour traiter des objets lisses. Notre modèle permet la subdivision multirésolution pour les objets rugueux. Nous avons déjà présenté une formule générale (1)

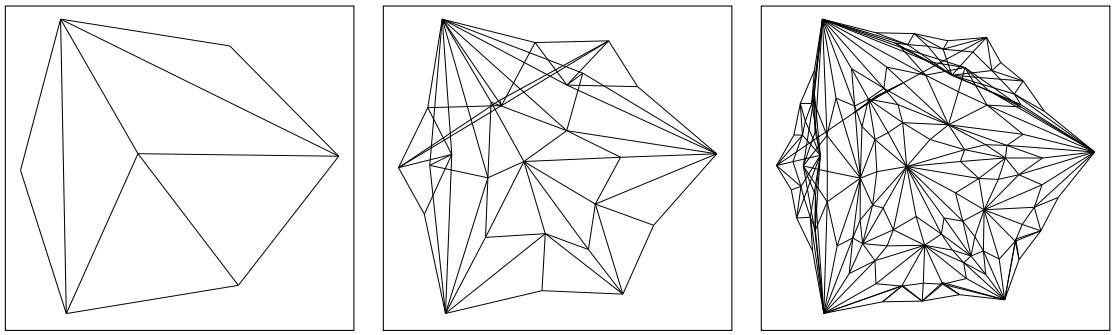


FIG. 39 – Exemple de subdivision de maillage triangulaire générés avec notre modèle de subdivision.

dans l'introduction , nous expliquons maintenant nos modèles en utilisant cette formule. Nous commençons en présentant le modèle qui considère la courbe, puis le cas de la surface sera traité.

3.3.1 Modèle des courbes

Dans ce cas $\mathcal{P}^J$ représente un ensemble ordonné de points $\mathcal{P}^J = (\mathcal{P}_i^J)_{i=0\ldots m-1}$ dans le plan $\mathbb{R}^2$ ou dans $\mathbb{R}^3$, le nombre minimum de points accepté par notre modèle est de quatre points. La règle d'augmentation du nombre de points est : si nous avons $m = 2^J+3$ points dans le niveau J alors nous aurons $m^{'} = 2^{J+1} + 3$ points dans le niveau $J + 1$, nous restreignons le nombre de points avec des numéros liés à des puissances de 2 afin de pouvoir appliquer l'analyse ainsi que la synthèse.

La définition de la somme $\oplus$ est la même que la définition (11) si nous considérons qu'un polygone est un ensemble ordonné de points. Le résultat de la fonction φ est un ensemble ordonné de points et pour le simplifier nous le notons par $\varphi = \varphi(\mathcal{P}^J)$. Cette fonction représente exactement notre opérateur de subdivision (3.7) pour la courbe présente dans la section précédente. Cette fonction fait simplement une étape de subdivision comme nous l'avons déjà faite avec notre opérateur de subdivision. Nous reformulons notre opérateur pour définir cette fonction φ comme ceci : si $\mathcal{P}^J$ a $m = 2^J+3$ points alors φ en a $m^{'} = 2^{J+1} + 3$. $\varphi(\mathcal{P}^J)$ est alors défini par :

$$\varphi(\mathcal{P}^J) = (\varphi_k)_{k=0\ldots 2^{J+1}+3}$$

et nous avons $\mathcal{P}^J = (\mathcal{P}_i^J)_{i=0\ldots m-1}$ où $m = 2^J + 3$ alors chaque point φ_k est défini par :

$$\varphi_k = \begin{cases} \frac{1}{2}(\mathcal{P}_i^J + \mathcal{P}_{i+1}^J) & \text{si } k = i = 0 \text{ ou } i = m-2 \text{ et } k = 2i \\ \mathcal{P}_i^J & \text{si } k = 2i-1 \text{ et } 0 < i < m-1 \\ \mathcal{M}(\mathcal{P}_i^J, \ldots, \mathcal{P}_{i+3}^J) & \text{si } k = 2i+2 \text{ et } i < m-3 \end{cases}$$

où $\mathcal{M}$ est un masque contenant dans ce cas une seule colonne et quatre lignes. Ce masque joue un rôle très important parcequ'il détermine la nature de la déformation appliquée sur une courbe par exemple une déformation fractale ou une déformation lisse.

Avant de définir la fonction ψ nous avons besoin de déterminer $\delta\mathcal{P}^J$. Le plus important ici est la taille de $\delta\mathcal{P}^J$: nous savons que $\mathcal{P}^J$ a $m = 2^J+3$ points alors $\delta\mathcal{P}^J$ en a $m-3 = 2^J$.

Maintenant si nous notons ψ de la même manière que φ, nous avons $\psi = \psi(\mathcal{P}^J, \delta\mathcal{P}^J)$ et $\psi = (\psi_k)_{k=0,\ldots,2^{J+1}}$ et chaque point ψ_k est défini par :

$$\psi_k = \begin{cases} 0 & \text{si } k = 0 \text{ ou } k = 2(m-1)-2 \\ 0 & \text{si } k = 2i-1 \text{ et } 0 < i < m-1 \\ \mathcal{R}(\mathcal{P}_{i+1}^J, \mathcal{P}_{i+2}^J)\delta\mathcal{P}_i^J & \text{si } k = 2i+2 \text{ et } i < m-3 \end{cases}$$

où $\mathcal{R}(\mathcal{P}_{i+1}^J, \mathcal{P}_{i+2}^J)$ est un repère que nous construisons en utilisant les deux points P_{i+1}^J et P_{i+2}^J comme ceci :

Soit P et Q deux points, le repère $\mathcal{R}(P,Q)$ est définis par :

$$\mathcal{R}(P,Q) = (\vec{u}, \vec{v}) \quad \text{où } \vec{u} = \frac{\overrightarrow{PQ}}{\|PQ\|} \text{ et } \vec{u} \perp \vec{v}$$

L'avantage d'une telle représentation du détail est la capacité de conserver la forme lorsque l'on applique des rotations ou des translations aux points de contrôle. Les étapes du calcul d'un nouveau point sont détaillées dans la figure 40. L'étape d'analyse est simple à réaliser. Elle consiste à conserver les points d'indice impair et d'utiliser ceux d'indice pair pour calculer le détail avec une inversion du repère local :

$$\delta\mathcal{P}_i^J = (\mathcal{R}(\mathcal{P}_{i+1}^J, \mathcal{P}_{i+2}^J))^{-1}(\mathcal{M}(\mathcal{P}_i^J, \ldots, \mathcal{P}_{i+3}^J)) \quad \text{avec } k = 2i+2 \text{ et } i < m-3$$

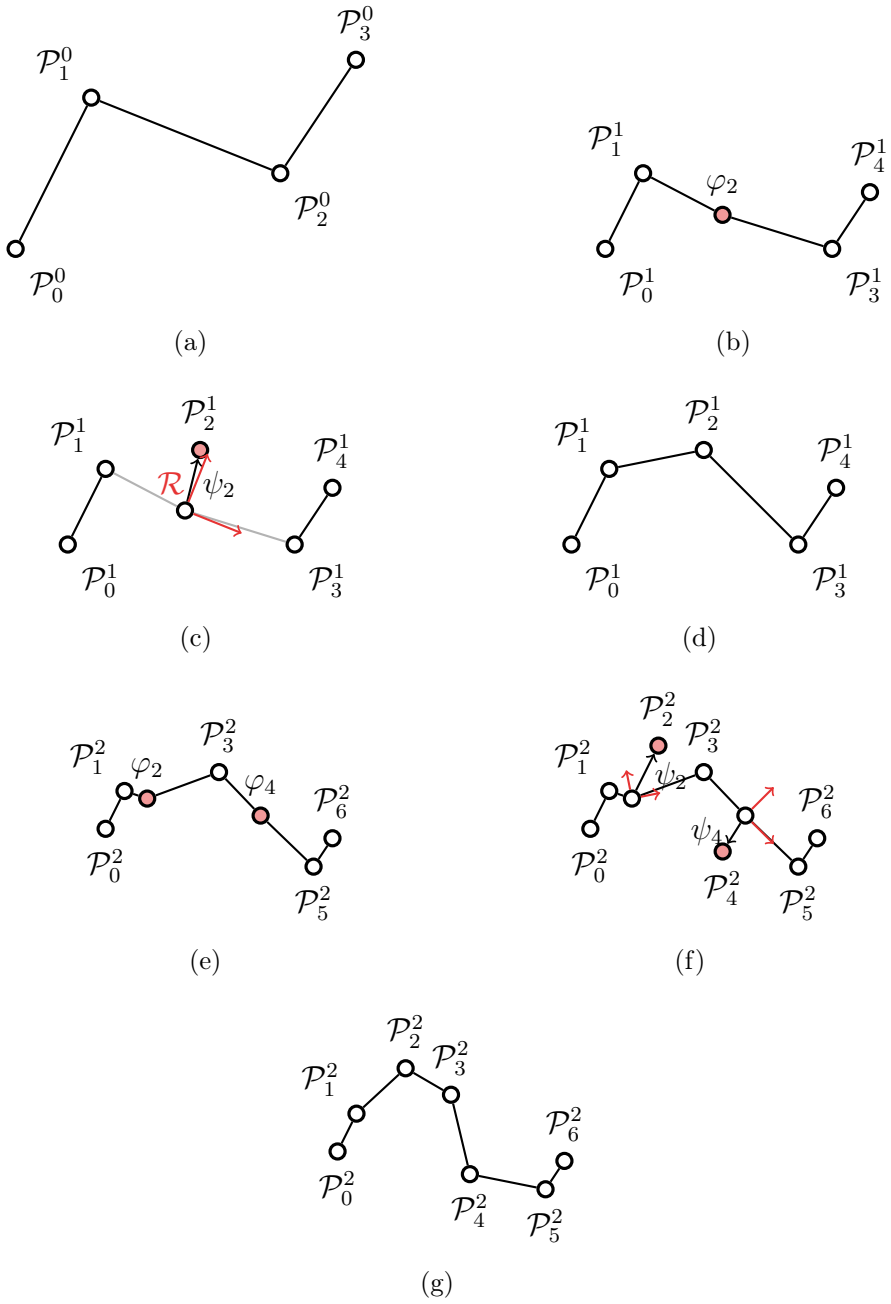


FIG. 40 – Les étapes de synthèse multirésolution :
(a) Les points initiaux au niveau $J = 0$.
(b) le Calcul de φ_2 en appliquant le masque.
(c) le détail ψ_2 est ajouté à φ_2.
(d) les nouveaux points au niveau $J = 1$.
(e) le Calcul de φ_2 et φ_4 en appliquant le masque.
(f) les détails ψ_2 et ψ_4 sont ajoutés à φ_2 et φ_4.
(g) les nouveaux points au niveau $J = 2$.

Pour les points extrêmes la formule sera inversée de cette manière :

$$\begin{cases} \mathcal{P}_0^J = 2 * \mathcal{P}_0^{J+1} - \mathcal{P}_1^J \\ \mathcal{P}_{m-1}^J = 2 * \mathcal{P}_{2*(m-1)-2}^{J+1} - \mathcal{P}_{m-2}^J \end{cases}$$

3.3.2 Modèle des surfaces

Dans cette section nous développons notre opérateur de subdivision pour une surface définie par une grille de points de contrôle vers un modèle de subdivision multirésolution. Dans ce cas $\mathcal{P}^J$ est un ensemble (grille) ordonné de points $\mathcal{P}^J = (\mathcal{P}_{i,j}^J)_{i=0...m-1,j=0...n-1}$ où $m = 2^U + 3$ et $n = 2^V + 3$ et $(J = min(U, V))$. La taille minimum de la grille acceptée par notre modèle est une grille ayant quatre lignes et quatre colonnes de points. La règle d'augmentation du nombre de données dans chaque ligne et chaque colonne de la grille est la suivante : Si nous avons une grille ayant $(2^U + 3) \times (2^V + 3)$ points dans le niveau J alors nous aurons une grille ayant $(2^{U+1} + 3) \times (2^{V+1} + 3)$ points dans le niveau $J + 1$. Nous restreignons le nombre de lignes et de colonnes de cette manière pour la même raison que nous avons mentionnée pour le cas de courbe c'est-à-dire pour pouvoir garantir que l'analyse est possible.

Nous commençons par définir la somme $\oplus$ introduite dans notre formule générale (1) dans l'introduction :

Définition 12 ***(la somme $\oplus$ entre deux grilles)*** *Soit $P = (P_{i,j})_{i=0...m-1,j=0...n-1}$ et $Q = (Q_{i,j})_{i=0,...m-1,j=0...n-1}$ deux grilles ordonnées de points ayant la même taille $m \times n$, nous définissions $P \oplus Q$ comme étant une nouvelle grille ordonnée ayant $m \times n$ points et nous la calculons comme ceci :*

$$P \oplus Q = (P_{i,j} + Q_{i,j})_{i=0...m-1,j=0...n-1}$$

Le résultat des fonctions φ et ψ est une grille de points. Si $\mathcal{P}^J$ a $(2^U + 3) \times (2^V + 3)$ points alors chacune de φ et ψ a $(2^{U+1} + 3) \times (2^{V+1} + 3)$ points où $(J = min(U, V))$. Il est simple de noter que nous utilisons le fait que chacune de φ et ψ est une grille seulement dans les définitions théoriques, dans l'implémentation nous employons une structure de donnée plus efficace en stockage et en temps de calcul.

Comme pour notre opérateur de subdivision nous utilisons plusieurs masques pour subdiviser la grille. En notant que $(h\delta\mathcal{P}_{i,j}^J, v\delta\mathcal{P}_{i,j}^J, d\delta\mathcal{P}_{i,j}^J)$ sont des vecteurs qui représentent respectivement les détails verticaux, horizontaux et diagonaux, et

$$\mathcal{R}_4 : (\mathbb{R}^3)^4 \longrightarrow (\mathbb{R}^3)^3$$

et

$$\mathcal{R}_6 : (\mathbb{R}^3)^6 \longrightarrow (\mathbb{R}^3)^3$$

sont des fonctions qui permettent d'automatiser le calcul de repères locaux en utilisant 4 ou 6 points. Le repère se compose de trois vecteurs perpendiculaires : deux d'entre eux sont dans le plan approximé qui minimise la distance aux 4 ou 6 points donnés, et le troisième est la normale à ce plan. Chaque étape de subdivision est alors accomplie en calculant la fonction φ en utilisant la formule (3.8). Nous définissons la fonction ψ comme ceci

- $\psi_{l,k} = 0$ pour :
  - Les quatre points de coins (les points oranges dans la figure 42) où $i\{0, m-2\}$, $j = \{0n-2\}$, $l = 2i$ et $l = 2i, k = 2j$;
  - Tous les points de bord horizontal avec des indices impairs de colonnes (les points gris dans la figure 42) où $i = \{0, m-2\}, 0 < j < n-1$ et $k = 2j-1, l = 2i$;
  - Tous les points de bord vertical avec des indices impairs de lignes (les points gris dans la figure 42) où $j = \{0, n-2\}, 0 < i < m-1$ et $l = 2i-1, k = 2j$;
  - Tous les points d'indices impairs de lignes avec des indices impairs de colonnes (les points noir dans la figure 42) où $0 < i < m-1$, $0 < j < n-1$ et $l = 2i-1, k = 2j-1$;
- Pour tous les points de bord horizontal avec des indices pairs de colonnes (les points verts dans la figure 42)

$$\psi_{l,k} = \mathcal{R}_4 \begin{pmatrix} \mathcal{P}^J_{i+1,j} & \mathcal{P}^J_{i+1,j+1} \\ \mathcal{P}^J_{i+2,j} & \mathcal{P}^J_{i+2,j+1} \end{pmatrix} h\delta\mathcal{P}^J_{i,j}$$

  où $i = \{0, m-2\}, j < n-3$ et $k = 2j+2, l = 2i$. Cette étape est expliquée dans la figure 41(c) ;
- Pour tous les points de bord vertical avec des indices pairs de lignes (les points verts dans la figure 42) :

$$\psi_{l,k} = \mathcal{R}_4 \begin{pmatrix} \mathcal{P}^J_{i,j+1} & \mathcal{P}^J_{i,j+2} \\ \mathcal{P}^J_{i+1,j+1} & \mathcal{P}^J_{i+1,j+2} \end{pmatrix} v\delta\mathcal{P}^J_{i,j}$$

  où $j = \{0, n-2\}, i < m-3$ et $l = 2i+2, k = 2j$. C'est exactement comme la figure 41(c) mais pour le bord vertical ;
- Pour tous les points d'indices impairs de lignes avec des indices pairs de colonnes

(les points bleus dans la figure 42) :

$$\psi_{l,k} = \mathcal{R}_6 \begin{pmatrix} \mathcal{P}^J_{i+1,j-1} & \mathcal{P}^J_{i+1,j} & \mathcal{P}^J_{i+1,j+1} \\ \mathcal{P}^J_{i+2,j-1} & \mathcal{P}^J_{i+2,j} & \mathcal{P}^J_{i+2,j+1} \end{pmatrix} h\delta\mathcal{P}^J_{i,j}$$

où $0 < i < m-1, j < n-3$ et $l = 2i-1, k = 2j+2$. La figure 41(b) montre comment nous pouvons calculer ce point.

- Pour tous les points d'indices pairs de lignes avec des indices impairs de colonnes (les points bleus dans la figure 42) :

$$\psi_{l,k} = \mathcal{R}_6 \begin{pmatrix} \mathcal{P}^J_{i-1,j+1} & \mathcal{P}^J_{i-1,j+2} \\ \mathcal{P}^J_{i,j+1} & \mathcal{P}^J_{i,j+2} \\ \mathcal{P}^J_{i+1,j+1} & \mathcal{P}^J_{i+1,j+2} \end{pmatrix} v\delta\mathcal{P}^J_{i,j}$$

où $0 < j < n-1, i < m-3$ et $l = 2i+2, k = 2j-1$;

- Finalement pour tous les points d'indices pairs de lignes avec des indices pairs de colonnes (les points rouges dans la figure 42) :

$$\psi_{l,k} = \mathcal{R}_4 \begin{pmatrix} \mathcal{P}^J_{i+1,j+1} & \mathcal{P}^J_{i+1,j+2} \\ \mathcal{P}^J_{i+2,j+1} & \mathcal{P}^J_{i+2,j+2} \end{pmatrix} d\delta\mathcal{P}^J_{i,j}$$

où $i < m-3, j < n-3$ et $l = 2i+2, k = 2j+2$. La figure 41(a) montre les points utilisés afin de calculer ce nouveau scalaire.

Nous utilisons plusieurs masques pour calculer le niveau $J+1$ à partir du niveau J pour avoir plus de contrôle sur la forme générée avec ces masques. Un autre avantage avec notre modèle est le fait que l'analyse (du niveau $J+1$ au niveau J) est effectuée rapidement en supprimant seulement des données et puis en calculant les détails associés à ces données.

La figure 42 représente une transition complète d'une grille (4×7) à une grille (5×11), toutes les boules noires de la grille droite ont des points ayant la même valeur que les boules correspondantes dans la grille de gauche.

Surface de profondeur :

Le cas de surface de profondeur est un cas particulier de notre modèle de surface, dans lequel les points et les vecteurs se composent d'une seule composante. Normalement les cordonnées pour chaque point dans la surface sont définies par : x est l'indice d'une colonne, y est l'indice d'une ligne et z est la valeur de scalaire de ce point. Dans ce cas notre grille de points de contrôle est une grille de scalaires de contrôle. tous les vecteurs de détails deviennent aussi des scalaires. Les détails dans ce cas sont donc seulement des

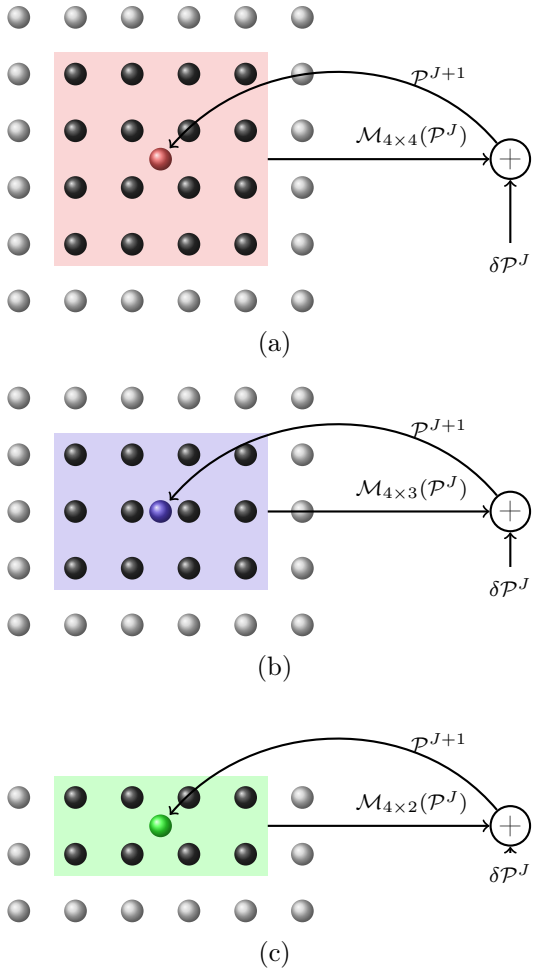


FIG. 41 – Principe de synthèse multirésolution dans le cas de la surface (transition du niveau J au niveau $J+1$).
(a) Un masque (4×4) est utilisé pour calculer les niveaux points (colonne paire, ligne paire).
(b) Un masque (4×3) est utilisé pour calculer les niveaux points (colonne paire, ligne impaire).
(c) Un masque (4×2) est utilisé pour calculer les niveaux points (colonne paire, première ligne).

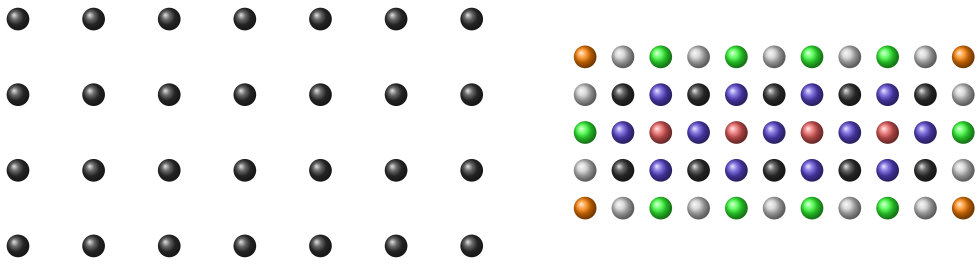


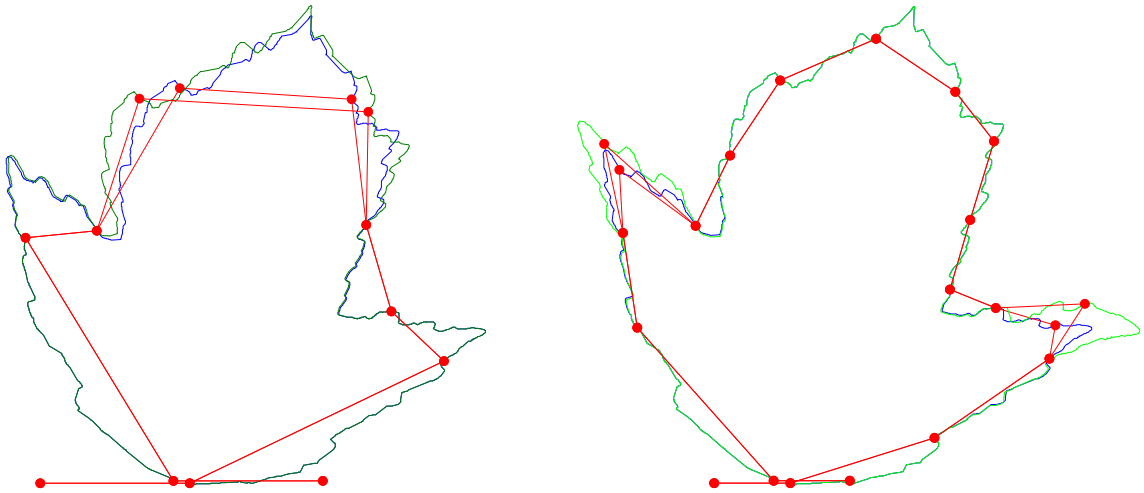
FIG. 42 – Transition complète du niveau J au niveau $J+1$.

déplacements sur l'axe z. Pour cette raison nous n'avons pas besoin de calculer les repères locaux parce que ces repères représentent un scalaire qui a la valeur 1.

3.4 Résultats

3.4.1 Courbes

Notre modèle est un modèle multirésolution qui est capable de représenter un objet à plusieurs échelles. Dans la figure 43 nous utilisons ce modèle afin de déformer localement la courbe représentant le contour d'une feuille. Une étape préliminaire d'analyse sur la courbe est nécessaire afin d'avoir les vecteurs de détails, puis nous pouvons déformer la courbe à n'importe quel niveau de résolution. Nous pouvons noter que la partie déformée de la courbe préserve ses détails locaux après la déformation.



(a) (b)

FIG. 43 – Édition multirésolution. (a) Avec 11 points de contrôle. (b) Avec 19 points de contrôle (l'étape suivante de 11 points de contrôle).

La déformation locale n'est pas le seul avantage de notre modèle. La figure 44 montre comment notre modèle peut garder les spécificités locales de la courbe grâce à la représentation relative du détail. La représentation des vecteurs de détails en utilisant un repère local garantit la préservation de la forme globale de la courbe même avec une forte déformation.

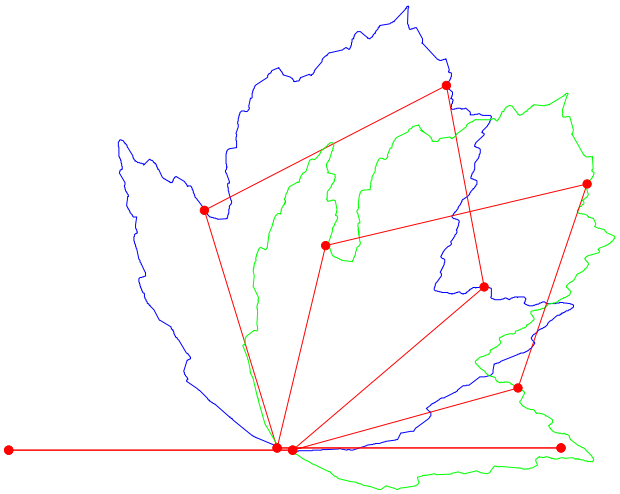


FIG. 44 – Conservation des details.

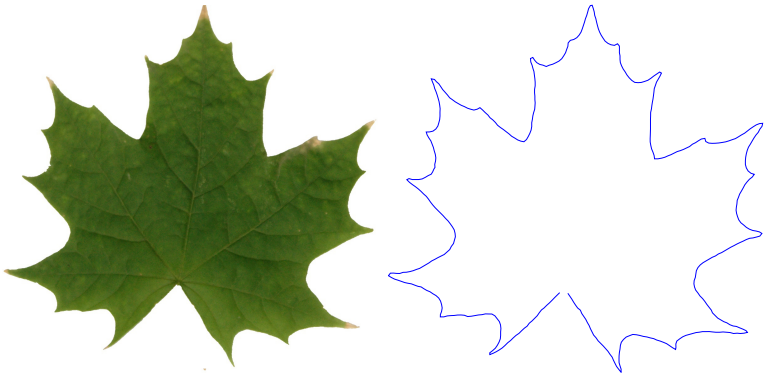


FIG. 45 – La feuille avec son contour.

3.4.2 Génération de variété de feuilles

Nous présentons dans cette section un exemple montrant l'utilisation de notre modèle pour générer une variété de feuilles à partir d'une image de feuille réelle. En commençant par une image de feuille réelle (figure 45 gauche), le contour de cette feuille est extrait (figure 45 droite). Ensuite nous échantillonnons le contour en prenant le nombre désiré de points ($515 = 2^9+3$ dans notre cas). Une étape d'analyse est appliquée sur les échantillons de la courbe du contour de cette feuille pour avoir les vecteurs de détails. Un niveau de résolution est choisi avec lequel chaque membre de la variété est généré (troisième niveau dans notre cas c'est-à-dire $2^3 + 1 = 11$ points de contrôle). Pour générer une famille de feuilles, nous choisissons deux états des points de contrôle : un état de début et un autre final. Ensuite nous générons le nombre désiré d'objets parmi les deux objets générés avec les points de contrôle de l'état de début et de l'état final. Une interpolation linéaire simple des points de contrôle initiaux et finals permet de générer la variété de feuilles présentées

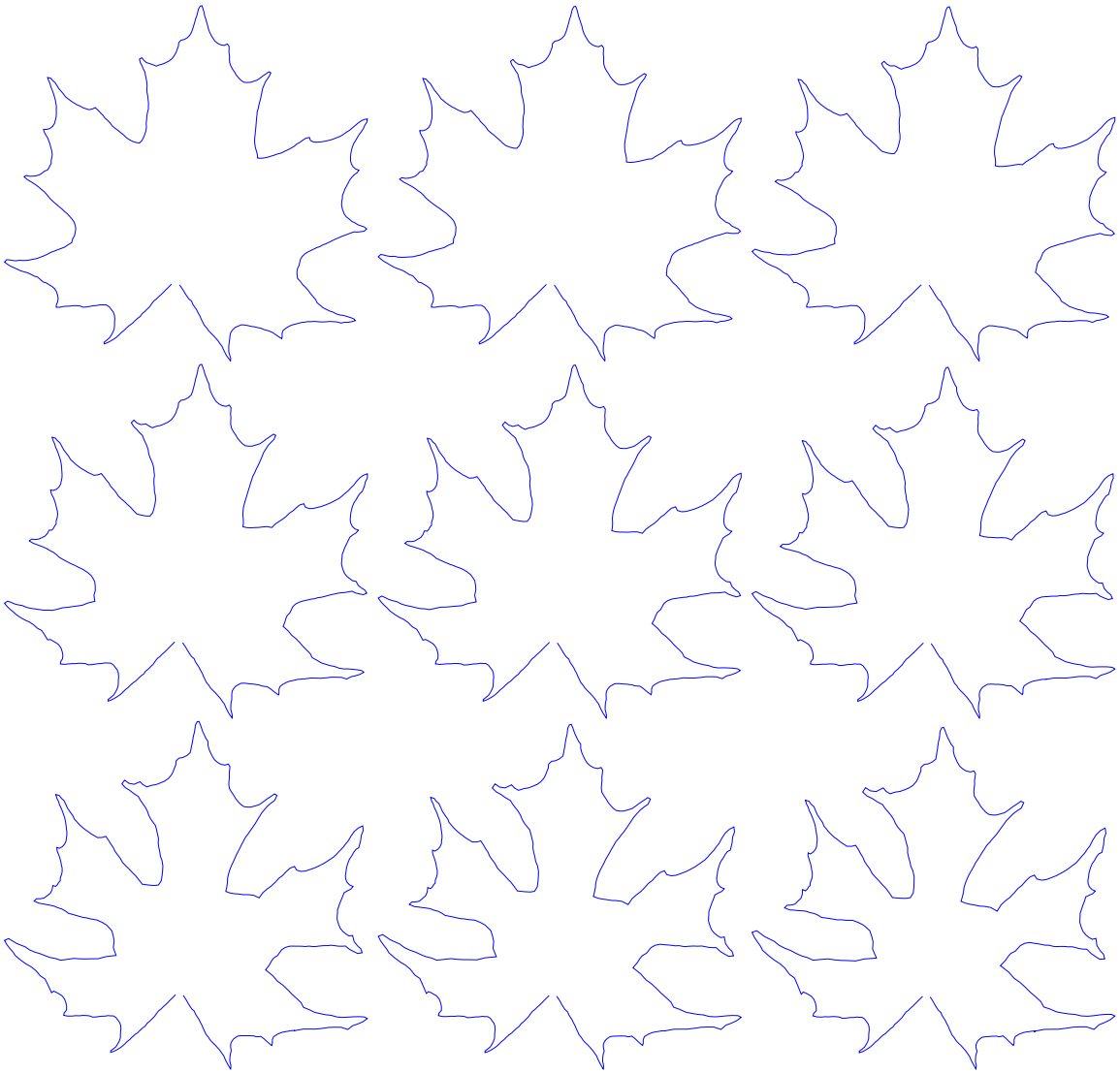


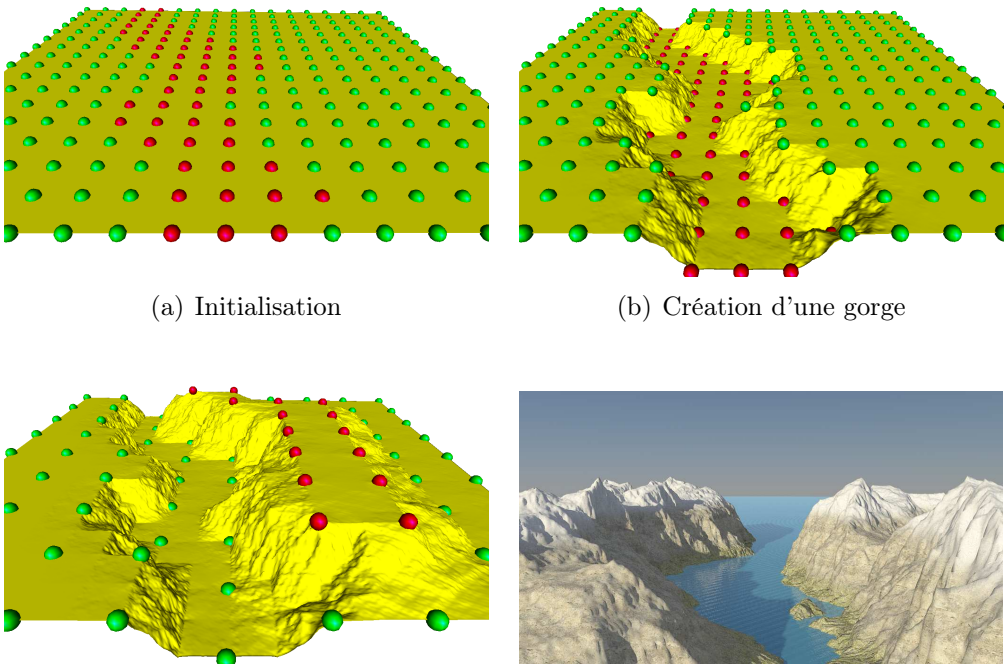
FIG. 46 – Une variété de feuilles

en figure 46. Nous pouvons bien noter que les détails du contour sont préservés et la déformation modifie seulement la forme globale de la feuille.

3.4.3 Surfaces de profondeur

Notre modèle fournit aussi un outil efficace de modélisation pour les surfaces définies par une grille de points. Dans cette section nous montrons l'efficacité de notre modèle en présentant des exemples de modélisation de terrain définis par une surface de profondeur (les cellules de la grille contiennent des scalaires de lieu de points). Dans la figure 47 nous montrons comment nous pouvons créer un terrain en utilisant notre modèle multirésolution. Nous commençons la modélisation en initialisant notre terrain comme une

surface de profondeur ayant une hauteur égale à zéro partout (figuer 47(a)). Puis nous l'analysons avec nos masques afin de trouver le modèle multirésolution associé (les points de contrôle, le détail). La figure 47(b) montre combien il est simple de créer une gorge interactivement en bougeant les points de contrôle. Nous avons aussi créé une montagne (figure 47(c)) en utilisant un niveau de résolution différent de celui utilisé pour la création de la gorge. L'avantage de modifier le terrain dans différentes résolutions est que nous pouvons contrôler la taille des caractéristiques. Par exemple, une caractéristique de petite taille, nous la créons dans un niveau raffiné de résolution. Par contre une caractéristique de grande taille, nous la créons dans un niveau grossier de résolution. La figure 47(d) montre le rendu de la forme finale du terrain.



(a) Initialisation (b) Création d'une gorge

(c) Ajout d'une montagne (d) Rendu de terrain

FIG. 47 – Étapes de modélisation de terrain de profondeur.

Notre modèle de subdivision est capable de générer des terrains réalistes. En bougeant seulement quelques points dans différents niveaux de résolution, des scènes réalistes sont produites. Les figures 48, 49 et 50 montrent des exemples de scènes naturelles modélisées avec notre modèle. Nous pouvons bien noter que plusieurs caractéristiques peuvent être modélisées par notre modèle (lac dans un environnement de montagne, chaînes de montagnes, canyon avec rivière, collines).

FIG. 48 – Une grande scène naturelle de lac (gauche) et de collines (droite) modélisées avec notre modèle de subdivision

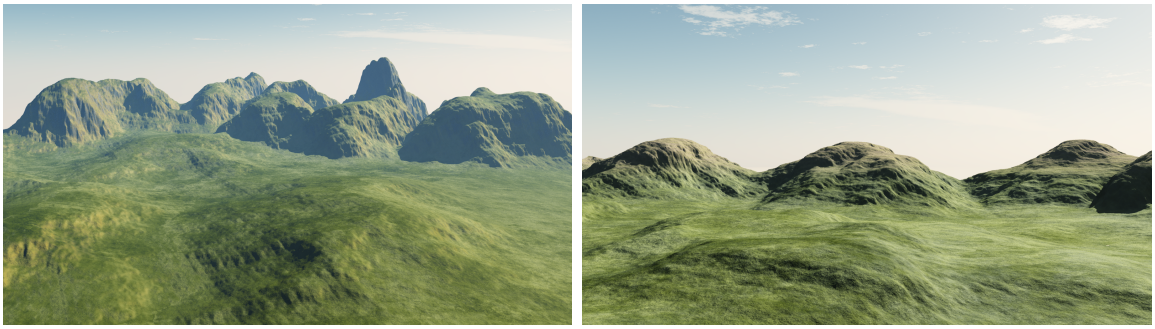


FIG. 49 – Des chaînes de montagnes modélisées avec notre modèle de subdivision

3.4.4 Surfaces tridimensionnelles

Notre modèle peut aussi construire des surfaces tridimensionnelles définies par une grille de points. Nous commençons par l'initialisation d'une surface dans l'exemple présenté dans la suite. Notre surface est définie comme ceci : x =l'indice de ligne, y =l'indice de colonne et $z = 0$. La figure 51 montre le terrain résultant. Nous pouvons bien noter que la surface est très accidentée grâce aux masques fractals.

3.5 Conclusion

Nous avons introduit dans ce chapitre un modèle de subdivision combiné avec une notion de détails. Contrairement aux modèles multirésolution connus qui traitent uniquement les objets lisses, ce modèle est capable de traiter et de générer des objets rugueux. Ce modèle nous permet de reconstruire et modéliser des courbes et des surfaces de manière efficace. Nous avons montré la capacité et la simplicité de notre modèle pour générer une variété d'objets. Notre modèle fournit un outil très efficace pour modéliser et déformer une surface définie par une grille de manière très réaliste.

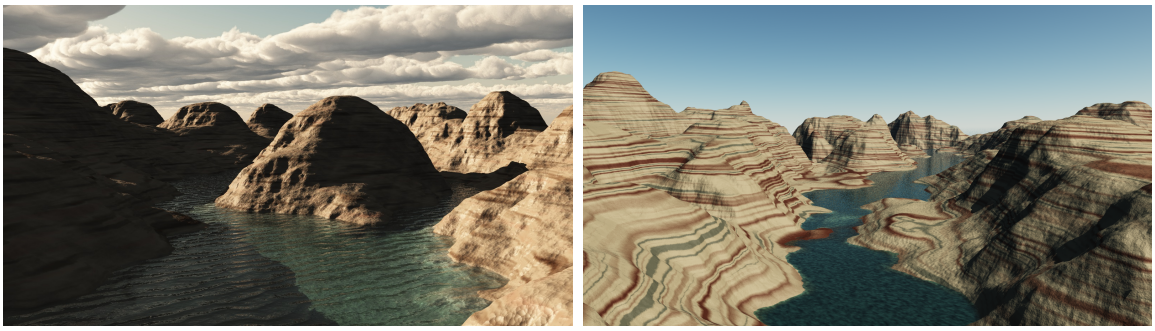
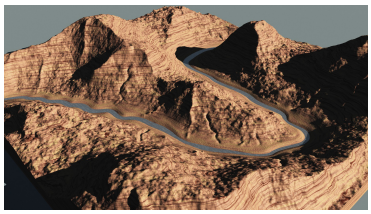
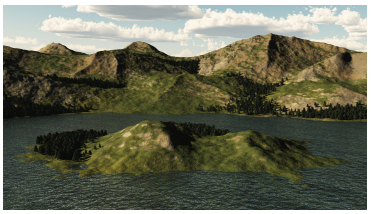
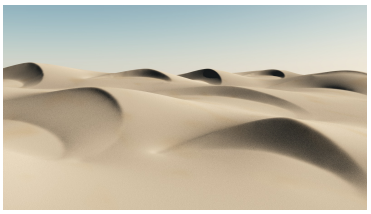


FIG. 50 – des canyons modélisés avec notre modèle de subdivision

FIG. 51 – Rendu de terrain tridimensionnel

Chapitre 4

Modèle de génération contrôlée de terrain

Sommaire

Dans ce chapitre nous introduisons un nouveau modèle contrôlable de génération de terrain. Nous contrôlons la génération en définissant un ensemble de courbes que nous appelons courbes caractéristiques. Des points de contrainte sont définis sur chaque courbe. Chacun de ces points est caractérisé par un ensemble de contraintes comme la hauteur, l'angle de pente, *etc.* Les autres parties du terrain sont générées automatiquement en utilisant des équations de diffusion en prenant les courbes caractéristiques comme des contraintes locales. La génération du terrain est faite interactivement grâce à une implémentation sur GPU[4]. Notre modèle de génération est un modèle déterministe, il donne le même résultat à chaque génération lorsque les courbes caractéristiques sont les mêmes. L'espace utilisé pour stocker les courbes est très compact car nous stockons seulement les points de contrôle et les points de contrainte pour chaque courbe. La description vectorielle du modèle permet notamment la génération à différentes résolutions de manière transparente.

Ce chapitre est organisé comme suit. Nous présentons dans la première section le modèle de diffusion, puis nous détaillons notre modèle de terrain dans la deuxième section, enfin nous expliquons les résultats obtenus en montrant l'efficacité de notre modèle pour générer des scènes réalistes.

4.1 Modèle de Diffusion

Notre modèle de génération est basé sur un modèle de diffusion contrôlé localement par un ensemble de courbes caractéristiques. Les modèles de diffusion sont connus depuis longtemps et ils sont utilisés pour résoudre plusieurs problèmes physiques, surtout les problèmes de diffusion de fluides et de chaleur. Dans cette section nous allons nous concentrer sur l'utilisation de ces modèles dans le domaine de l'informatique graphique, et notamment le traitement d'image.

4.1.1 Diffusion et traitement d'image

La diffusion employant l'équation de Poisson a été largement utilisée dans la vision par ordinateur. Elle se pose tout naturellement comme une condition nécessaire à la solution de certains problèmes de variations. Dans le domaine du traitement d'image plusieurs applications d'édition d'images ont utilisé la diffusion afin de modifier globalement ou localement une image.

Dans [EG01], un système est mis en place pour modifier un image via un ensemble

[4] *Graphic Processing Unit*

clairsemé de ses éléments de bordure. Pour supprimer un objet, les arêtes associées sont supprimées ; pour ajouter un objet, les arêtes associées ainsi que les valeurs de couleurs des deux côtés de chacune de ces arêtes sont incorporées. La nouvelle image est alors obtenue par une interpolation lisse des couleurs associées à la nouvelle série d'arêtes. Cette méthode revient à résoudre une équation de Laplace avec des conditions aux bords de Dirichlet données par les couleurs autour des arêtes.

Fattal et al. [FLW02], présentent une méthode dans laquelle le champ de gradient d'une image est remis en échelle de façon non linéaire, produisant un champ vectoriel qui n'est plus un champ de gradient. Une nouvelle image est alors obtenue en résolvant une équation de Poisson avec la divergence de champ vectoriel comme côté droit de l'équation de Poisson et sous les conditions de bords de Neumann précisant que la valeur du gradient de la nouvelle image dans la direction de normale au bords est de zéro. Une méthode d'interpolation générique basée aussi sur la résolution des équations de Poisson est introduite dans [PGB03]. Une variété de nouveaux outils est présentée pour l'édition sans couture des régions d'une image. Le premier ensemble d'outils permet l'importation de deux régions opaques et transparentes de l'image source dans une région de destination. La deuxième série est basée sur un formalisme similaire et permet à l'utilisateur de modifier l'apparence d'une image de façon transparente, dans une région choisie. Ces changements peuvent affecter la texture, l'éclairage et la couleur des objets situés dans la région, ou de faire une sélection rectangulaire. Dans [MP08] un logiciel d'édition d'image est présenté, qui permet aux artistes de peindre dans le domaine du gradient en temps réel les informations sur des images de taille d'un megapixel. Une brosse des arêtes est introduite, conçue pour la sélection des arêtes et la relecture. Ces brosses, qui sont couplées à des modes particuliers de fusion, permettent aux utilisateurs de réaliser l'éclairage global et le contraste en utilisant uniquement des manipulations d'images locales.

Bien que ces méthodes offrent des outils très efficaces pour l'édition d'images, elles utilisent une représentation des pixels pour représenter leurs primitives, qui ne nous convient pas dans le contexte de la génération de terrains.

Dans [OBW+08], une nouvelle primitive représentée de manière vectorielle est introduite afin de créer des images lisses. Cette primitive est appelée courbe de diffusion. Une courbe de diffusion divise l'espace à travers lequel elle est établie, définissant des couleurs différentes de chaque côté. Ces couleurs peuvent varier de manière lisse le long de la courbe. En outre, les régions de transition des couleurs d'un côté de la courbe à l'autre peuvent être contrôlées. Étant donné un ensemble de courbes de diffusion, l'image finale est construite en résolvant une équation de Poisson dont les contraintes sont spécifiées par l'ensemble des gradients à travers toutes les courbes de diffusion. Comme toutes

les primitives vectorielles, les courbes de diffusion supportent commodément une variété d'opérations, y compris l'édition basée sur la géométrie, l'animation, et la stylisation.

Notre modèle de diffusion est basé sur le travail de Orzan et al. [OBW+08] et notamment leur représentation vectorielle des primitives. Notre modèle se différencie du leur par la définition de contraintes le long de la courbe et l'introduction de contraintes de gradient.

4.1.2 Base théorique

Dans cette section nous présentons la base théorique de la diffusion. Comme notre modèle est basé sur celle de la diffusion utilisée dans le domaine du traitement d'image, nous commençons par expliquer ce genre de diffusion. Afin d'expliquer les modèles de diffusion utilisés en traitement d'image, nous devons commencer par les équations aux dérivées partielles.

Équations aux dérivées partielles

Les équations aux dérivées partielles sont au cœur d'un grand nombre de systèmes physiques, comme les fluides, les champs électromagnétiques, le corps humain, etc...

Dans la littérature mathématique, les équations aux dérivées partielles (EDPs) sont classées en trois catégories [PFTV93b], hyperbolique, parabolique et elliptique. L'exemple classique d'une équation hyperbolique est l'équation des ondes unidimensionnelle :

$$\frac{\partial^2 F}{\partial t^2} = v^2 \frac{\partial^2 F}{\partial x^2}$$

où $v = constante$ est la vitesse de propagation des ondes. L'exemple classique de l'équation parabolique est l'équation de diffusion :

$$\frac{\partial F}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial F}{\partial x}\right)$$

où D est le coefficient de diffusion. L'exemple classique des équations elliptiques est l'équation de Poisson :

$$\frac{\partial^2 F}{\partial x^2} + \frac{\partial^2 F}{\partial y^2} = \rho(x, y)$$

où le terme de source ρ est donné. Si ρ est égal à zéro, cette équation devient l'équation de Laplace. Nous pouvons écrire cette équation également comme ceci :

$$\Delta F = \rho(x, y)$$

où $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ est l'opérateur de Laplace.

Le dernier genre d'équations (Poisson et Laplace) est le genre qui est largement utilisé dans le domaine du traitement d'image. La divergence d'un champs de gradient G sera généralement utilisée comme le coté droit de cette équation. Dans notre modèle nous employons aussi ce genre d'équations en la développant selon nos besoins.

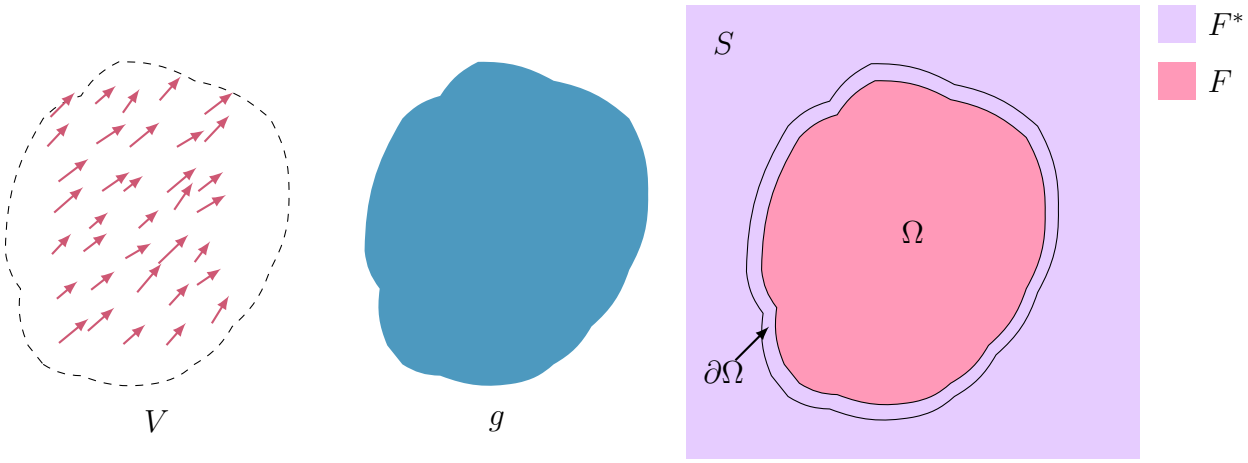


FIG. 52 – Diffusion guidée par une champ vectoriel [PGB03]

La diffusion des images guidée par un champs de gradient

Dans ce paragraphe, nous détaillons l'interpolation d'image à l'aide d'un champ de vecteurs d'orientation [PGB03]. Soit S, un sous-ensemble fermé de $\mathbb{R}^2$, le domaine de définition d'une image scalaire, et soit Ω un sous-ensemble fermé de S avec frontière $\partial\Omega$. Soit F^* une fonction scalaire connue, définie sur S moins l'intérieur de Ω et soit F une fonction scalaire inconnue définie sur l'intérieur de Ω. Enfin, soit V un champ de vecteurs défini sur Ω. (Figure 52)

La solution la plus simple qui interpole F sur Ω est définie comme la solution du problème de minimisation :

$$\min_F \int\int \mid \nabla F \mid^2 \text{ avec } F \mid_{\partial\Omega} = F^* \mid_{\partial\Omega}$$

où $\nabla = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right]$ est l'opérateur de gradient. La minimisation doit satisfaire l'équation d'Euler-Lagrange associée :

$$\Delta F = 0 \text{ sur } \Omega \text{ avec } F \mid_{\partial\Omega} = F^* \mid_{\partial\Omega}$$

La dernière equation est l'équation de Laplace avec conditions de bords de Dirichlet. Pour les applications d'édition d'images, cette méthode simple produit un résultat insatisfaisant, une interpolation floue. Cela peut être surmonté en utilisant une variété de méthodes. Une méthode parmi celles-ci consiste à modifier le problème en introduisant des contraintes supplémentaires sous la forme d'un champ d'orientation, comme expliqué ci-dessous.

Un champ d'orientation est un champ vectoriel V utilisé dans une version étendue du problème de minimisation présenté au dessus :

$$\min_F \int\int \mid \nabla F - V \mid^2 \text{ avec } F \mid_{\partial\Omega} = F^* \mid_{\partial\Omega}$$

la solution de ce problème est la solution unique de l'équation de Poisson avec des conditions de bords de Dirichlet :

$$\Delta F = divV \text{ sur } \Omega, \text{ avec } F \mid_{\partial\Omega} = F^* \mid_{\partial\Omega}$$

où $divV = \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$ est la divergence de $V = (u, v)$. Au lieu de la divergence d'un champ vectoriel, nous pouvons utiliser une fonction g qui peut être une fonction calculée à partir d'une autre image. C'est le mécanisme fondamental de l'édition d'images en couleur utilisant l'équation de Poisson : trois équations de Poisson de la forme présentée sont résolues indépendamment dans les trois canaux de couleur RGB.

Dans notre modèle de diffusion, nous employons cette idée afin de diffuser les paramètres contrôlant la génération de terrain. Un autre type d'équation sera utilisé pour bien contrôler la solution en cas de gradient nul.

4.1.3 Notre modèle de diffusion

Le modèle de diffusion utilisé dans notre générateur de terrain est basé sur l'équation de Poisson avec des contraintes locales. Dans notre générateur nous rencontrons parfois un cas où la caractéristique de terrain oblige à traiter un gradient nul. Dans le cas de l'équation de Poisson, un gradient nul la transforme en équation de Laplace. Pour cette raison nous n'utilisons pas l'équation de Poisson mais nous employons une combinaison de l'équation de Laplace et une équation particulière aux premières dérivées partielles.

Notre modèle de diffusion emploie trois types d'équations : l'équation aux dérivées d'ordre 2 de Laplace, une équation particulière aux dérivées d'ordre 1 pour traiter le gradient, et une identification (ordre 0) pour appliquer les contraintes locales.

L'équation d'ordre 2 indique que le Laplacien d'un champ F est tout simplement nul :

$$\Delta F = 0 \tag{4.1}$$

où F est un champ scalaire défini sur $\mathbb{R}^2$. Cette équation est utilisée pour exprimer que le champ est lisse.

Quand nous voulons guider la diffusion dans la direction du gradient, nous utilisons une équation d'identification d'ordre 1 que nous appelons l'équation de gradient :

$$\nabla F = \mathbf{G} \tag{4.2}$$

où $\mathbf{G} = (G_x, G_y)$ est un champ vectoriel que nous voulons imposer.

Lorsque nous voulons mettre directement une contrainte sur la valeur de F, nous utilisons directement une équation d'identification d'ordre 0 :

$$F = \mathcal{F} \tag{4.3}$$

Comme nous voulons un contrôle précis sur le résultat, plusieurs équations peuvent être combinées au même point.

4.1.4 Solveur de multigrille

Il y a plusieurs techniques pour résoudre les équations aux dérivées partielles. Dans cette section nous nous concentrons sur les solutions numériques de ces équations. La solution la plus célèbre de ce type d'équations est de les transformer en système linéaire d'équations puis de résoudre ce système en utilisant une méthode parmi les méthodes existantes. Comme toutes les méthodes de diffusion citées dans cette section, nous choisissons une méthode de relaxation afin de résoudre notre système linéaire. Plus précisément nous employons la relaxation de Jacobi qui est résolue avec la technique de multigrille.

Discrétisation du modèle

Afin de pouvoir résoudre notre modèle numériquement, nous devons le discrétiser. Pour cette raison nous utilisons les opérateurs numériques suivants :

- L'opérateur de Laplace Δ que nous le discrétisons comme ceci :

$$\begin{aligned}\Delta F &= \frac{\partial^2 F}{\partial x^2} + \frac{\partial^2 F}{\partial y^2} \\ &= \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} * \begin{pmatrix} 0 & F(x+1,y) & 0 \\ F(x,y-1) & F(x,y) & F(x,y+1) \\ 0 & F(x-1,y) & 0 \end{pmatrix}\end{aligned}$$

  où $A * B = \sum_{i,j} A_{i,j} B_{i,j}$
- L'opérateur de gradient ∇ est discrétisé comme ceci :

$$\nabla F = (\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}) = (F(x,y) - F(x+dx,y), F(x,y) - F(x,y+dy))$$

  où $dx, dy = \pm 1$.

La discrétisation est faite sur une grille régulière.

> **Remarque :** Nous expliquons la méthode multigrille en utilisant une grille carrée mais l'utilisation d'une telle grille n'est pas obligatoire, nous pouvons simplement traiter la méthode dans le cas d'une grille rectangulaire.

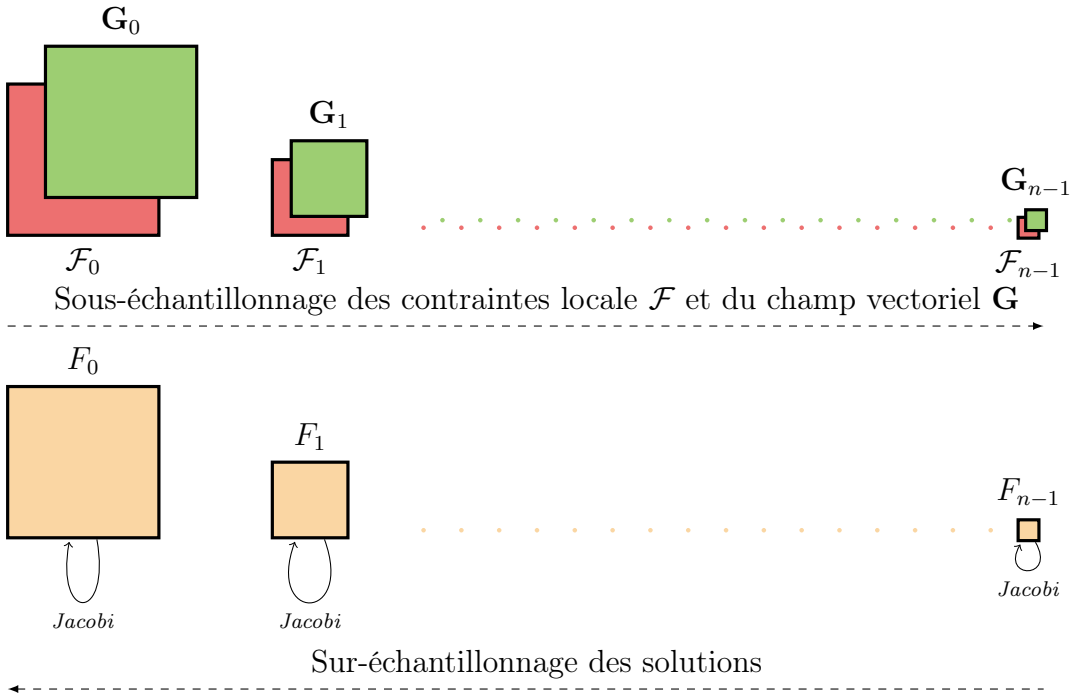
La méthode multigrille

La méthode multigrille consiste à résoudre une version grossière du système afin d'obtenir les composantes de basse fréquence de la solution. Ensuite le domaine est raffiné pour obtenir les composantes de hautes fréquences. Nous utilisons la relaxation de Jacobi afin de calculer la solution dans chaque niveau de la multigrille. (Figure 53). La méthode se compose des étapes suivantes :

- Sous-échantillonnage de la grille des contraintes locales. Nous sous-échantillonnons cette grille en prenant en compte seulement les cellules voisines qui sont des cellules de contraintes. (Les rectangles rouges dans la figure 53).
- Sous-échantillonnage de la grille du champ vectoriel. Nous utilisons l'opérateur suivant pour accomplir cette étape (Les rectangles verts dans la figure 53) :

$$\begin{vmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{vmatrix}$$

- Nous résolvons notre système linéaire pour chaque niveau de la multigrille en



$\mathbf{G}_i$: représentent les champs vectoriels dans les différente résolutions .
$\mathcal{F}_i$: représentent les contraintes locales dans les différente résolutions .
F_i : représentent les solutions du système linéaire dans les différente résolutions .

FIG. 53 – Solution de l'équation de Diffusion en utilisant le solveur multigrille.

utilisant la relaxation de Jacobi. Nous distinguons plusieurs cas :

- Le point (*x*,*y*) présente une contrôle strict. Dans ce cas la valeur est celle de la contrainte (équation d'identification) :

$$F^{k+1}(x,y) = F^k(x,y) = \mathcal{F} \tag{4.4}$$

- Le point (*x*,*y*) présente une contrainte de gradient mais pas de contrainte stricte. Dans ce cas, la valeur est calculée à partir du gradient mais aussi avec une combinaison de l'équation de Laplace :

$$\begin{aligned} F^{k+1}(x,y) &= \lambda(cos^2(\gamma)(F^k(x+dx,y)+G_x(x,y)) \\ &+ sin^2(\gamma)(F^k(x,y+dy)+G_y(x,y))) \\ &+ \frac{1-\lambda}{4}\left(F^k(x+1,y)+F^k(x,y+1)+F^k(x-1,y)+F^k(x,y-1)\right) \end{aligned} \tag{4.5}$$

où γ est l'angle de la normale à la courbe caractéristique dont est issue la contrainte (Section 4.2.2).

- Le point (x,y) ne présente aucune contrainte. L'equation de Laplace est utilisée :

$$F^{k+1}(x,y) = \frac{1}{4}\left(F^k(x+1,y) + F^k(x,y+1) + F^k(x-1,y) + F^k(x,y-1)\right) \tag{4.6}$$

- La solution obtenue pour chaque niveau est sur-échantillonnée pour l'utiliser comme une solution de départ dans le niveau le plus raffiné en employant l'opérateur :

$$\begin{vmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{vmatrix}$$

Remarque : Le nombre d'itérations utilisé dépend du niveau traité dans le multigrille. Nous utilisons beaucoup d'itérations dans les niveaux grossiers de la multigrille tandis qu'un nombre moindre d'itérations est utilisé dans les niveaux raffinés afin de gagner du temps du calcul.

Implémentation sur GPU

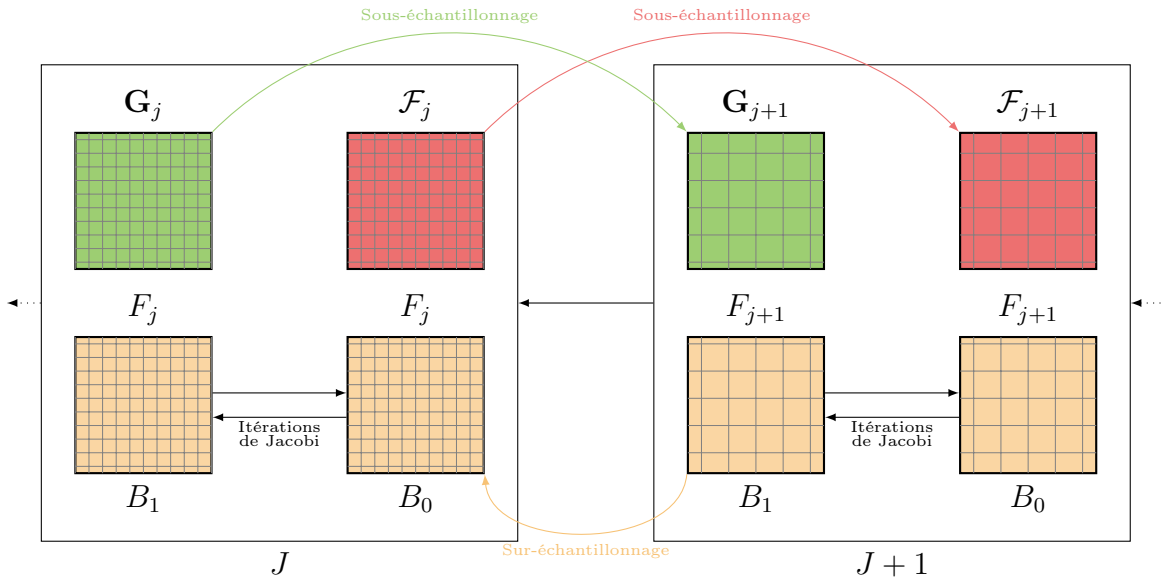


FIG. 54 – Deux niveaux successifs du solveur de multigrille. Chaque grille est représentée par une texture sur la mémoire de GPU. La relaxation de Jacobi est accomplie en utilisant deux textures B_0 et B_1.

Comme les méthodes de diffusion citées ci-dessus, notre méthode utilise un solveur de

multigrille pour résoudre le modèle de diffusion avec une utilisation intensive des capacités du GPU[4]. Plusieurs articles présentent de telles techniques pour exploiter les GPU dans ce contexte [GWL+03, BFGS03]. Dans ce paragraphe nous expliquons l'implémentation de notre solveur en utilisant la capacité du GPU.

Pour pouvoir utiliser le GPU, il faut stocker toutes nos données sur des images de texture pour qu'elles soient accessibles par le GPU. Notre implémentation se compose des étapes suivantes (Figure 54) :

- Nous copions la carte des contraintes locales sur une texture et nous faisons de même pour la carte du champ vectoriel de gradient.
- Nous sous-échantillonnons les textures des contraintes $\mathcal{F}$ locales et du champ vectoriel G jusqu'à arriver à la version la plus grossière de chacune de ces texture en stockant chaque version des textures.
- Nous utilisons deux textures (B_0 et B_1 dans la figure 54) pour accomplir une itération de Jacobi. Si B_0, B_1 sont la source et la destination respectivement dans l'itération k, alors B_0, B_1 seront inversées dans l'itération $k+1$ pour éviter des copies de texture.
- Après avoir raffiné la solution F dans le niveau $J+1$ de multigrille, nous la sur-échantillonnons au niveau J en l'utilisant comme une solution de départ (Figure 54).
- Une fois terminé le raffinement au niveau $J+1$ nous libérons toutes les textures utilisées.

4.2 Modèle de terrain

Dans cette section nous expliquons notre modèle de terrain. Ce modèle est défini en trois étapes principales : la définition des courbes caractéristiques, la rastérisation de ces courbes sur deux images de texture, et finalement la diffusion des paramètres. Dans la première étape chaque courbe caractéristique est associée à une ensemble de points de contrainte, un ensemble de paramètres est attaché à chacun de ces points. La deuxième étape pixellise (*rasterize*) ces paramètres en utilisant deux images de texture, une pour les paramètres de terrain (hauteur, rugosité) et une autre pour les paramètres de gradient. La dernière étape génère le terrain en diffusant ces paramètres. (Figure 55)

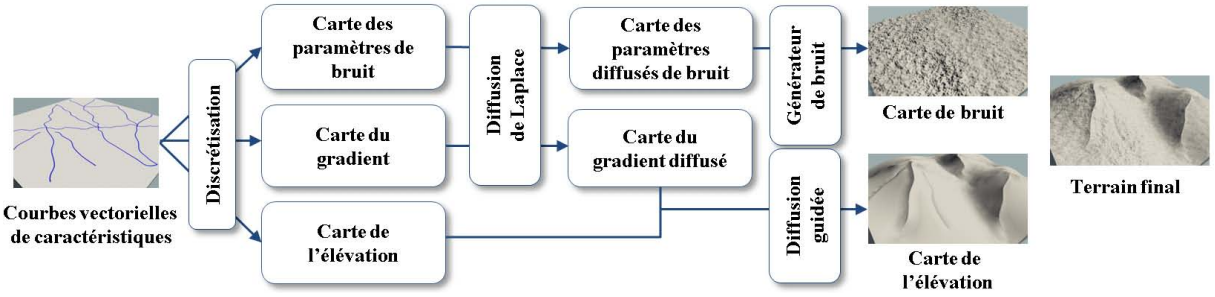


FIG. 55 – Vue d'ensemble des étapes du modèle de terrain

4.2.1 Primitives de modélisation de terrain

Dans cette section, nous présentons les courbes de contrôle qui sont utilisées pour décrire et contrôler la forme des reliefs de la scène. Il y a une grande variété de reliefs dans la nature, ce qui les rend difficiles à modéliser de façon cohérente. Nous proposons d'utiliser les courbes de contrôle vectorielles comme primitive générique de modélisation afin de représenter une grande variété de caractéristiques terrestres, y compris les lignes de crête simples ou multiples, des lits de rivières, collines, falaises et crevasses. Étant donné un ensemble de courbes de contrôle avec les paramètres correspondants de contrainte d'élévation , la pente et le bruit qui leur sont rattachés, les processus global de générations s'accompli comme suit (Figure 55) :

- Le calcul des cartes de contraintes de l'altitude, la pente, l'amplitude du bruit et la rugosité en appliquant une étape de rastérisation.
- La diffusion des contraintes de bruit, de pente et d'élévation à l'aide de la diffusion de Laplace.
- Le calcul d'une carte d'élévation lisse en utilisant une diffusion guidée contrôlée par la carte de gradient diffusée.
- La génération de la carte de détails du terrain en utilisant les paramètres de bruit diffusés.

La carte finale d'élévation qui représente le terrain est obtenue en ajoutant la carte lisse d'élévation et la carte de détails.

Courbes caractéristiques

Les courbes caractéristiques, notées C_k, sont des courbes vectorielles définies comme des courbes cubiques de B-Spline en dimension deux (Figure 56). Nous attachons des points de contrainte P_i à la courbe C_k. Chaque point de contrainte Pi est défini par les

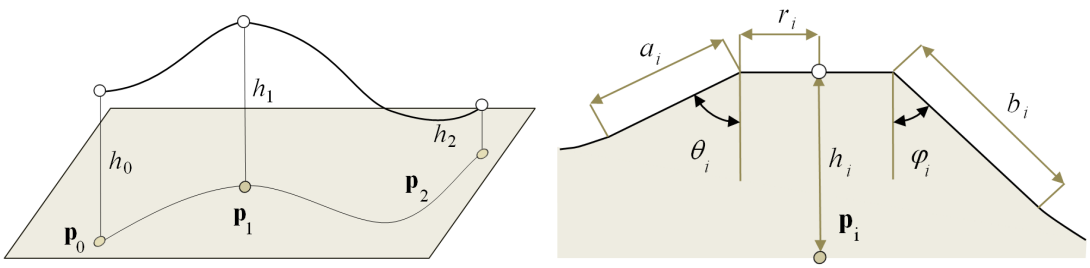


FIG. 56 – Courbe de contrôle et les contraintes géométriques associées(altitude et angle)

attributs suivants :

$$\mathcal{P}_i = ((h_i, r_i), (a_i, b_i, \theta_i, \varphi_i), (\mathcal{A}_i, \mathcal{R}_i), u_i)$$

où $i = 1 \ldots m$ et $m = |\mathcal{P}|$. u_i est la coordonnée linéaire le long de la spline.

Il n'est pas obligatoire de joindre tous les types de contraintes à une courbe. Les contraintes sont regroupées en trois catégories :

- Les contraintes d'altitude (h_i, r_i).
- Les contraintes d'angle $(a_i, b_i, \theta_i, \varphi_i)$.
- Les contraintes de bruit $(\mathcal{A}_i, \mathcal{R}_i)$.

θ_i et φ_i se réfèrent à l'angle de pente alors que a_i et b_i dénotent la longueur de la contrainte de pente des deux côtés de la courbe (Figure 56). h_i représente la contrainte de hauteur et r_i dénote le rayon du plateau des deux côtés de la courbe caractéristique. Des contraintes de bruit sont utilisées afin de contrôler les paramètres du générateur de bruit. Notre générateur de bruit est contrôlé par deux paramètres : l'amplitude notée $\mathcal{A}$ et la rugosité notée $\mathcal{R}$.

L'utilisateur peut choisir de joindre une seule catégorie, ou deux ou toutes les catégories de contraintes à la courbe. La figure 56 montre un exemple d'une courbe caractéristiques définie dans le plan (x, y) avec quatre points de contrôle. Le premier et le dernier points de contrôle de la courbe sont des points de contrainte et nous avons mis un autre point de contrainte sur la courbe à $u = 0,3$. La courbe de B-Splines est utilisée pour définir la projection d'une ligne de crête ou un lit de rivière sur le plan (x, y) tandis que les contraintes sont utilisées afin de définir les paramètres d'élévation, de gradient et de bruit le long de cette courbe.

Deux points de contrainte au moins sont attachés à une courbe, aux extrémités. Entre les points de contrainte, chaque attribut est interpolé linéairement le long de la courbe, sauf pour l'élévation h_i qui est interpolée avec une interpolation cubique pour éviter les lignes de crête linéaires qui ne sont pas naturelles.

Modélisation de terrain

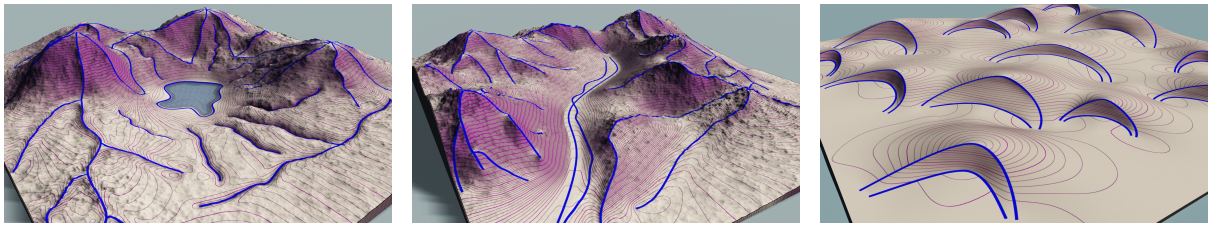


FIG. 57 – Différents types de paysages (lac, montagne et désert) créées avec 45, 59 et 26 courbes de contrôle respectivement.

Les courbes caractéristiques fournissent à l'utilisateur un outil générique et efficace de modélisation pour définir et contrôler les reliefs du terrain (figure 57). Nous montrons dans la suite l'efficacité de notre modèle en présentant plusieurs caractéristiques naturelles modélisées grâce à nos courbes caractéristiques.

Ligne de crête : Une ligne de crête peut être obtenue par une seule courbe caractéristique (Figure 58 gauche). Les contraintes d'élévation indiquent les altitudes sur la ligne de crête, tandis que les contraintes d'angle décrivent le degré d'inclinaison de chaque côté de la courbe. Une petit rayon de contrainte d'altitude est souvent choisi dans ce cas.

Lignes de crête multiples : Parfois, une montagne n'a pas seulement une seule ligne de crête. Il est possible de combiner des lignes de crête multiples qui ont un point commun de sommet avec la même contrainte d'élévation et dont la direction et les contraintes d'angle de pente sont différentes. Un exemple de montagne avec trois lignes de crête se trouve dans la figure 58 droite.

Lit de rivière : Un lit de rivière peut également être obtenu par une seule courbe caractéristique (Figure 59 gauche). Les contraintes d'élévation sont dans un petit intervalle et le rayon de contraintes d'élévation indique la largeur du lit de la rivière. Les contraintes d'angle de pente sont légèrement vers le haut ($> 90°$) avec un petit rayon associé.

Falaise : Une falaise est formée d'une seule ligne de caractéristiques combinant deux contraintes d'angle d'inclinaison qui sont radicalement opposées de chaque côté de la courbe(Figure 59 droite).

Colline : Une colline est obtenue en mettant des angles horizontaux comme contraintes d'angle de la pente de chaque côté de la courbe caractéristique (Figure 60 gauche). Ces

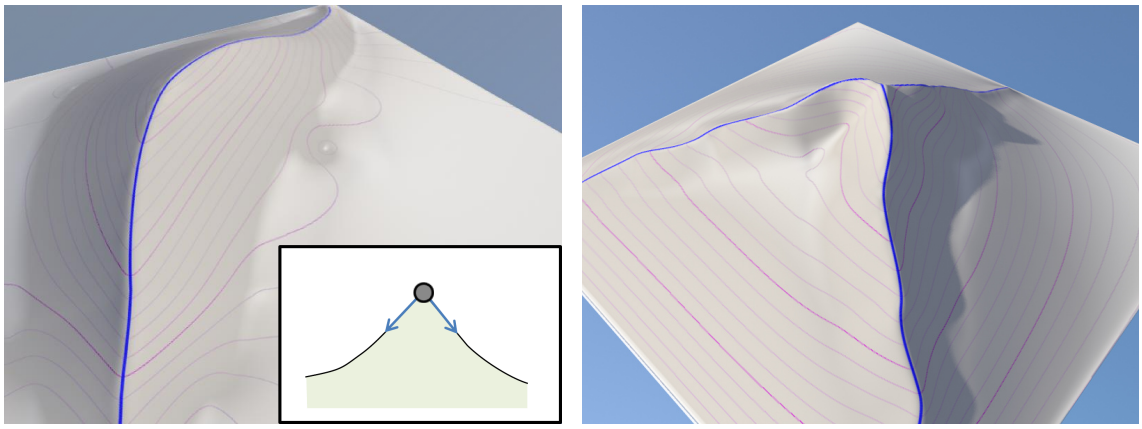


FIG. 58 – Les courbes caractéristique de ligne de crête (gauche) et de lignes de crête multiples (droite)

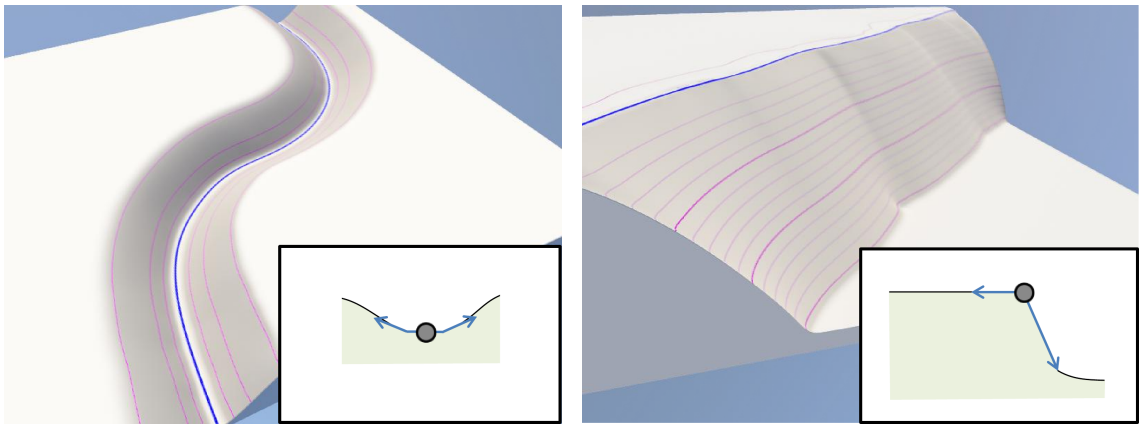


FIG. 59 – Les courbes caractéristique de lit de rivière (gauche) et de falaise (droite)

contraintes impliquent un gradient nul et imposent donc aux points dans le voisinage d'être à la même altitude.

Crevasse : Une crevasse est obtenue en mettant les deux angles de pente (gauche et droit) vers le haut ($> 90°$) comme contraintes d'angle de chaque côté de la courbe caractéristique(Figure 60 droite). Dans ce cas, il n'est pas nécessaire d'attacher les contraintes strictes d'élévation.

Contrôle de bruit : Deux paramètres permettent de définir des contraintes sur l'amplitude et la rugosité du bruit. La figure 61 montre l'influence du paramètre d'amplitude (gauche) et de rugosité (droite). Dans cet exemple, les contraintes de bruit ont été seulement utilisés.

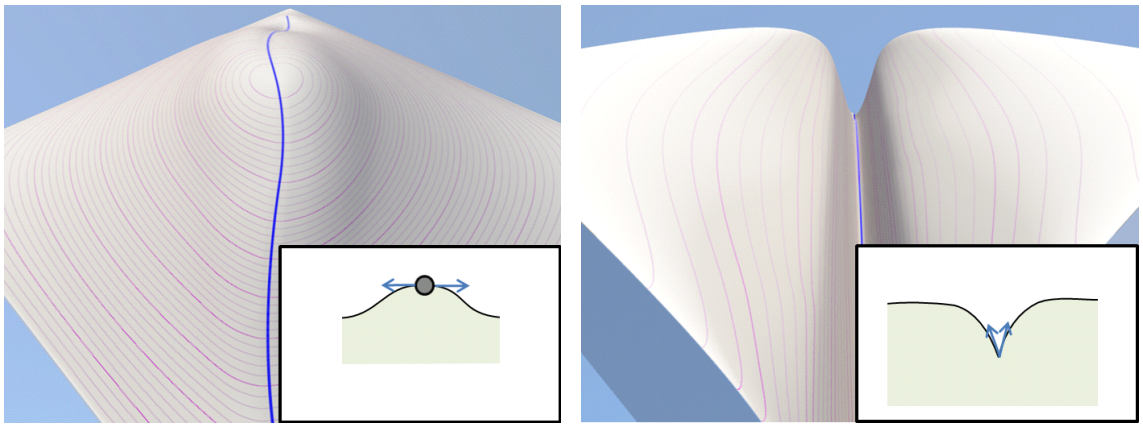


FIG. 60 – Les courbes caractéristiques d'une colline (gauche) et d'une crevasse (droite)

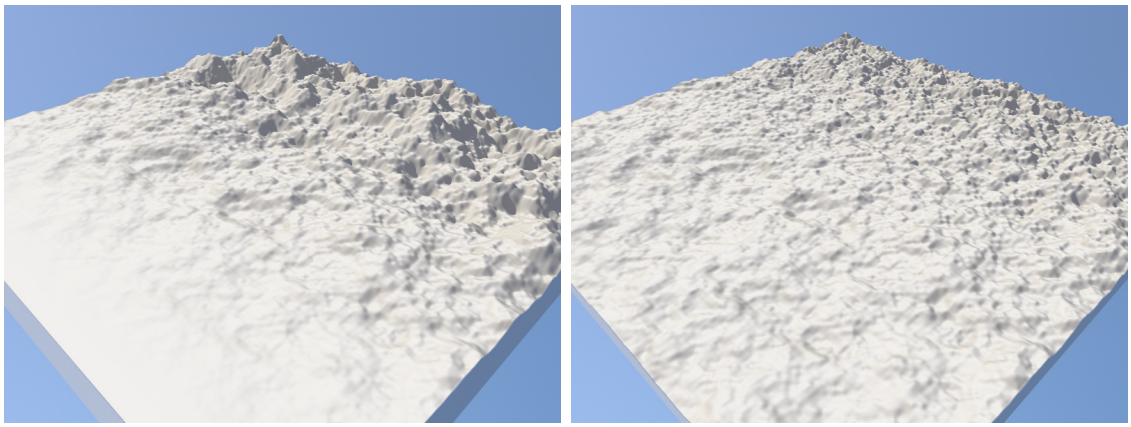


FIG. 61 – Le contrôle des paramètres d'amplitude (gauche) et de rugosité (droite) de bruit

4.2.2 L'algorithme de génération de terrain

Dans cette section, nous présentons la génération du champ de hauteur à partir de l'ensemble des courbes caractéristiques définies par l'utilisateur. Plusieurs étapes doivent être accomplies afin de générer le champ de hauteur final du terrain. Premièrement nous pixellisons (*rasterize*) les courbes sur deux images de texture dont les couleurs représentent les différents paramètres attachés aux courbes. Après avoir pixellisé (*rasterized*) les courbes nous appliquons deux passes de diffusion, la première diffuse les paramètres de gradient afin de remplir les régions d'intersection et la deuxième passe diffuse la hauteur en la guidant par le gradient, ainsi que les paramètres de bruit. Enfin les paramètres de bruit sont utilisés pour générer une carte de bruit que nous ajoutons au champ de hauteur, le résultat de cette addition représente le terrain final.

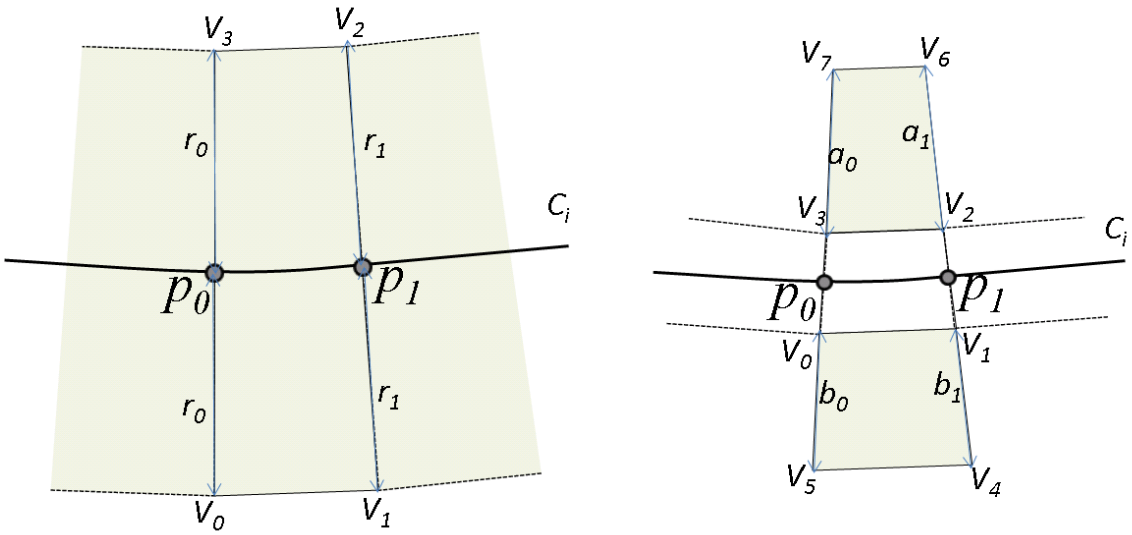


FIG. 62 – (À gauche) les quadrangles représentants les contraintes strictes. (V_0, V_1, V_2, V_3) sont les sommets d'un quadrangle. Les couleurs pour V_0 et V_3 sont ($R = h_0$, $G = \mathcal{A}_0$, $B = \mathcal{R}_0$) et pour V_1 et V_2 sont ($R = h_1$, $G = \mathcal{A}_1$, $B = \mathcal{R}_1$). (À droite) les quadrangles représentants les contraintes du gradient. Deux quadrangles sont utilisés (V_3, V_2, V_6, V_7) à gauche et (V_5, V_4, V_1, V_0) à droite. Les couleurs des sommets proches de la courbe sont calculées à partir des angles (θ, φ) et de la normale à la courbe et les couleurs de sommets éloignés de la courbe sont mis à zéro.

Rastérisation

Les courbes caractéristiques définies par l'utilisateur sont des graphiques vectoriels, elles sont indépendantes de la résolution de l'image. La méthode de diffusion s'appuie sur une représentation rastérisée des données. Par conséquent, un pré-traitement de rastérisation est obligatoire pour représenter les contraintes (élévation, paramètres de bruit, angle de pente) par des images.

La courbe est d'abord discrétisée en parties linéaires. À chaque coordonnée linéaire, l'ensemble des paramètres est interpolé linéairement à partir des points de contraintes à l'exception de la contrainte d'élévation où une interpolation cubique est utilisée pour éviter les lignes de crête linéaire qui ne sont pas naturelles sur les montagnes. Puis, de chaque côté de la courbe, des quadrangles sont générés dans la direction de la normale dont les longueurs sont définies à partir des valeurs de rayons interpolés r, a et b.

Les couleurs des sommets du quadrangle correspondent aux valeurs interpolées de contraintes (Figure 62). Pour des contraintes strictes, les quadrangles sont peints avec des valeurs uniformes de h, $\mathcal{A}$ et $\mathcal{R}$. Pour les primitives des angles de pente, la couleur du sommet est mise à sa valeur interpolée correspondant le long de la courbe et est mise à zéro

à l'extrémité du quadrangle. De cette façon, nous évitons les discontinuités de gradient et les artefacts qui en résultent. L'information de gradient est représentée comme une combinaison de la direction de la normale à la courbe caractéristique et la norme du gradient. Ces valeurs sont inversées lorsque la courbe caractéristique ne contient pas de contraintes strictes d'élévation. De cette façon, la propagation du gradient se comporte mieux et nécessite moins d'itérations dans le solveur de multigrille.

Le problème d'intersection.

Les courbes caractéristiques peuvent être placées n'importe où de telle sorte qu'elles peuvent se croiser. L'intersection des contraintes strictes (paramètres de bruit et d'élévation) peut être résolue automatiquement par la moyenne de toutes les valeurs de contraintes. Toutefois, l'intersection du gradient est un problème plus complexe en raison de gradients contradictoires qui peuvent avoir des directions complètement opposées. Ce problème est encore plus difficile lorsque le nombre d'intersections de régions de gradient augmente. Afin d'avoir une solution générique à ce problème, nous laissons les régions

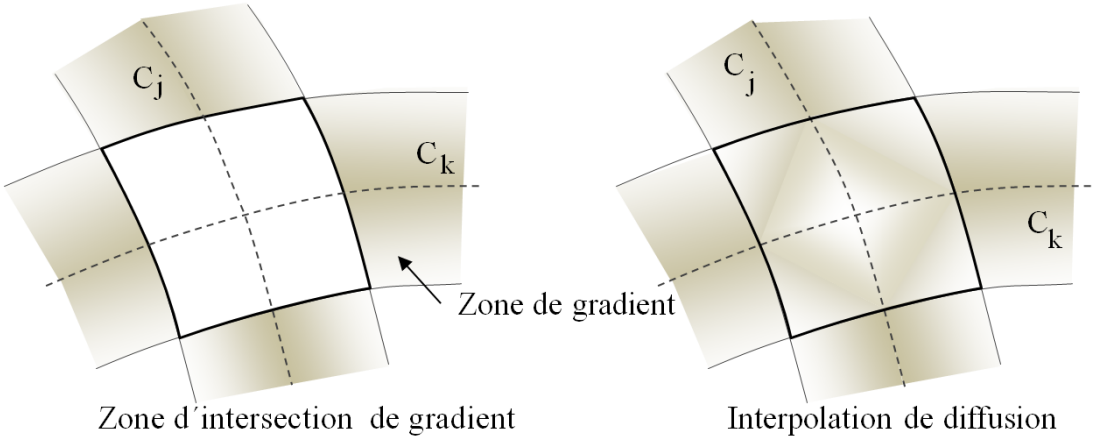


FIG. 63 – La solution du problème d'intersection de gradient.

d'intersection vides et effectuons une diffusion de Laplace pour remplir les trous. La figure 63 illustre le problème d'intersection lorsque deux courbes caractéristiques C_j et C_k se croisent. La méthode de diffusion tend naturellement à lisser le champ de gradient (Figure 63 à droite) et élimine les discontinuités.

Diffusion

Notre méthode de diffusion est basée sur les travaux de Orzan et al.[OBW+08]. Ils introduisent une représentation vectorielle des images lisses, avec l'utilisation des courbes

de B-Spline comme primitives. Chacune de ces courbes divise l'espace et les contraintes de couleur sont définies de chaque côté de la courbe. Les gradients sont utilisés pour effectuer une transition lisse entre ces contraintes le long de la courbe. À l'inverse, dans notre méthode les contraintes sont définies le long de la courbe et les gradients font des transitions lisses entre les contraintes et les autres zones du terrain.

Comme nous l'avons expliqué dans la section sur la théorie de la diffusion, plusieurs types d'équations aux dérivées partielles sont employées par notre modèle de terrain : Laplace (4.1) , gradient (4.2) et identification (4.3). La première est utilisée quand il n'y a pas d'informations de gradient. La deuxième guide la diffusion par un champ vectoriel de gradient. Le dernier type d'équation applique une contrainte stricte locale.

Les raisons pour lesquelles nous décidons d'utiliser une équation d'ordre 1 (équation de gradient (4.2)) au lieu de l'équation de Poisson pour guider la diffusion par un champ vectoriel de gradient sont :

- Cela permet d'éviter le calcul de la divergence du champ gradient.
- Une contrainte de gradient nul transforme l'équation de Poisson en équation de Laplace avec laquelle nous perdons l'influence de gradient.
- À cause de cette limitation nous ne pouvons pas générer de colline avec une équation de Poisson.

Comme nous l'avons déjà dit, pour résoudre les équations de diffusion, nous avons besoin d'abord de les discrétiser sur une grille rectangulaire de dimension 2. À cette fin nous avons utilisé un système d'intégration d'Euler (4.5). Dans cette formule nous avons besoin de choisir les valeurs de dx et dy. Afin d'accomplir ce choix, nous utilisons la direction de la normale pour choisir la bonne valeur pour dx et dy. Nous utilisons cette technique car il faut garantir la récupération de la valeur la plus proche de la courbe caractéristique. La formule (4.6) est utilisée afin de résoudre l'equation de Laplace.

Comme nous l'avons déjà dit, pour avoir un contrôle strict sur le résultat, plusieurs équations peuvent être concernées au même point. Ce système sur-contraint est résolu en utilisant une combinaison de termes de relaxations de Jacobi dans le solveur itératif.

Génération de bruit

Tout type de générateur de bruit peut être utilisé dans notre système à condition qu'il puisse être paramétré par deux champs scalaires : l'amplitude $A(x, y)$ et la rugosité $R(x, y)$. L'application du paramètre de rugosité directement sur un générateur de bruit de Perlin aux paramètres de la fréquence et la taille de caractéristique brise la continuité et donne des effets artificiels circulaires. Dans notre méthode, nous générons un bruit

multifractal basé sur un bruit de Perlin $S(x, y)$:

$$N(x, y) = A(x, y) \sum_{k=0}^{n} \frac{1}{r^{k(1-R(x,y))}} S(r^k x, r^k y)$$

où $n+1$ détermine le nombre d'octaves et R est la lacunarité. Dans notre implémentation, nous utilisons une génération multifractale de bruit en 4 octaves ($n = 3$ et $r = 2$). Notre bruit de Perlin est basé sur un générateur procédural déterministe de bruit. Le coefficient de rugosité R n'influence pas les différentes échelles de la combinaison multifractale, il ne change que la manière dont ils sont mélangés. Quand nous mettons $R = 1$, les différentes échelles sont mélangées de manière égale, tandis que des valeurs plus faibles de R feront affaiblir les coefficients de mélange des fréquences plus élevées. Les paramètres A et R du générateur de bruit sont remplacés par nos paramètres diffusés $\mathcal{A}$ et $\mathcal{R}$ respectivement.

> **Remarque :** La lacunarité est une mesure de la façon dont une fractale remplit l'espace. Il est utilisé pour classer plus finement les fractales et les textures qui apparaissent visuellement très différentes tout en partageant la même dimension fractale. Les fractales denses ont une lacunarité faible et si la grossièreté de fractales augmente, la lacunarité augmente aussi.

4.2.3 Détails d'implémentation

Afin d'atteindre des performances de génération interactives, l'implémentation utilisée dans notre générateur fait un usage intensif du GPU. Nous stockons nos données sur des images de textures afin de pouvoir employer la capacité de parallélisme introduite par le GPU moderne. Dans cette section nous expliquons comment nous représentons nos données avec des textures et les différents *shaders* que nous utilisons. Nous présentons aussi les techniques utilisées pour implémenter notre méthode, et comment nous avons résolu les différents problèmes apparus durant la mise en œuvre de notre générateur.

Représentation des données

Après avoir rastérisé les courbes caractéristiques, toutes les données résultantes de la rastérisation peuvent être représentées par des images. Cette idée est très importante lorsque nous cherchons à utiliser le GPU car ces images peuvent être traitées comme des textures par les shaders de fragment.

Comme nous l'avons déjà expliqué, nous utilisons les couleurs de sommets durant l'étape de rastérisation pour stocker les contraintes. Après cette étape de rastérisation

nous avons plusieurs cartes de contraintes qui peuvent être multiplexées sur une seule texture à l'aide des différentes composantes. Les paramètres des contraintes d'élévation et de bruit sont rassemblés dans les composantes RGB d'une seule image de texture RGBA. La dernière composante alpha de la texture est utilisée pour indiquer les contraintes qui ont été mises sur les différentes régions.

Les composantes x et y de la direction de la normale à la courbe caractéristique et la norme du gradient sont stockés dans les composantes rouge, vert et bleu d'une autre image de texture RGB.

L'implémentation de multigrille

Nous utilisons un algorithme multigrille pour résoudre le système linéaire obtenu par discrétisation. La méthode de [OBW+08] est utilisée et adaptée à notre problème pour résoudre la multigrille. Cet algorithme est calculé en temps interactif pour une grille carrée à une gamme de résolution de 512^2 à 2048^2 à l'aide des capacités de GPU. Comme nous l'avons déjà présenté, la méthode de multigrille consiste à résoudre une version grossière du système et obtenir les composantes de basse fréquence de la solution, puis à raffiner le domaine afin d'obtenir les composantes de haute fréquence. La relaxation de Jacobi est utilisée pour résoudre chaque niveau de multigrille, et nous limitons le nombre d'itérations de relaxation pour atteindre de bonnes performances.

Soient $0 \leq \alpha \leq 1$, $0 \leq \beta \leq 1$ et $0 \leq \alpha + \beta \leq 1$. Le terme de relaxation de Jacobi est calculé par une combinaison pondérée des trois types d'équations :

$$F^{k+1}(x,y) = \alpha F_L^{k+1}(x,y) + \beta F_G^{k+1}(x,y) + (1-\alpha-\beta)F_I^{k+1}(x,y)$$

Où chaque terme est calculé comme suit :

1. L'équation de Laplace $F_L^{k+1}(x,y)$. Cette équation est localement linéarisée en utilisant la formule (4.6).
2. L'équation de gradient $F_G^{k+1}(x,y)$. La norme du gradient est ajoutée au voisinage à la direction de la normale en employant la formule (4.5). L'angle γ dans cette équation est calculé en utilisant les composantes x et y de la normale à la courbe caractéristique.
3. Identification. Il s'agit de la simple équation :

$$F_I^{k+1}(x,y) = \mathcal{F}(x,y)$$

Parfois, les points de la grille peuvent être concernés par plusieurs équations simultané-

ment. Par exemple, dans les régions de contraintes de gradient, nous avons fixé $\alpha = \beta = \frac{1}{2}$. De cette façon, les itérations de Jacobi tendent à satisfaire l'équation de la pente et l'équation de Laplace dans le même temps. Dans les régions de contraintes strictes d'élévation, nous avons mis $\alpha = \beta = 0$. Si nous voulons approximer (et non pas interpoler) les contraintes d'élévation, nous pouvons prendre $\alpha > 0$ et la solution sera plus lisse. Mais l'approximation adoucit les arêtes caractéristiques et n'est pas convenable dans notre cas. Partout ailleurs, nous avons mis $\alpha = 1$ et $\beta = 0$.

Cinq champs scalaires doivent être calculés avec l'algorithme standard de la diffusion de Laplace : l'amplitude du bruit $\mathcal{A}$, la rugosité du bruit $\mathcal{R}$, la norme du gradient et les composantes x et y de la direction de la normale à la courbe caractéristique. Le champ scalaire qui représente l'altitude h est calculé en utilisant un algorithme de diffusion guidée. Les valeurs de α et β sont déterminées par les courbes caractéristiques et les positions de contraintes.

D'abord, nous diffusons la norme du gradient et les deux composantes de la direction de la normale à la courbe caractéristique afin de remplir les lacunes dans les régions intersectées du gradient. Ensuite, le gradient diffusé est utilisé pour diffuser l'altitude h. Les paramètres de bruit $\mathcal{R}$ et $\mathcal{A}$ sont diffusés dans le même *shader* pour la performance, car ils sont algorithmiquement identiques. La seule différence est que la diffusion guidée doit calculer le voisin dans la direction du gradient. L'altitude et les paramètres de bruit sont multiplexés sur les différentes composantes d'une seule image de texture.

Génération de bruit

Le signal d'entrée sur lequel ce générateur est basé est un bruit de Perlin interpolé bi-linéairement qui est généré procéduralement sur le GPU. Afin de garantir que ce signal est déterministe et donne toujours le même résultat sur une entrée donnée (x, y), nous calculons le nombre pseudo-aléatoire grâce à des opérations de bits sur x et y.

4.3 Résultats

Nous avons implémenté notre méthode dans une application de modélisation codée en $C++$. Toutes les images montrées à travers ce chapitre ont été créées à l'aide de cette application. Les rendus ont été réalisés en utilisant *Mental-Ray* sur des maillages texturés procéduralement que nous avons produits avec notre technique.

4.3.1 Réalisme

Notre méthode crée des paysages réalistes et se compare favorablement en termes d'efficacité et de qualité aux techniques existantes d'édition [GMS09, RME09]. La principale raison à cela est que notre approche fournit un bon contrôle sur les caractéristiques du relief. En outre, les courbes de contrôle sont un outil générique qui peut être utilisé pour créer une grande variété de différents types de terrains, comme des îles volcaniques (Figure 65), des montagnes et des paysages de désert (Figure 66), des canyons avec des lits de rivières complexes (Figure 69) et même un paysage de collines (Figure 67).

Modèle	Caractéristiques	Taille
Montagne	59	2,89
Lac	45	2,26
Canyon	74	4,48
Colline	55	2,25
Île volcanique	48	2,95
Désert	26	1,30

TAB. 1 – Statistiques de terrain : le nombre de courbes caractéristiques et la taille de la structure de données(en kilo-octets).

Modèle	Temps de calcul		
Résolution de grille	512^2	1024^2	2048^2
Montagne	0,187	0,270	0,670
Lac	0,185	0,261	0,690
Canyon	0,216	0,339	0,811
Colline	0,188	0,266	0,611
Île volcanique	0,168	0,275	0,675
Désert	0,171	0,253	0,635

TAB. 2 – Temps de calcul (en secondes) pour la génération du terrain à la résolution de 512^2, 1024^2 et 2048^2.

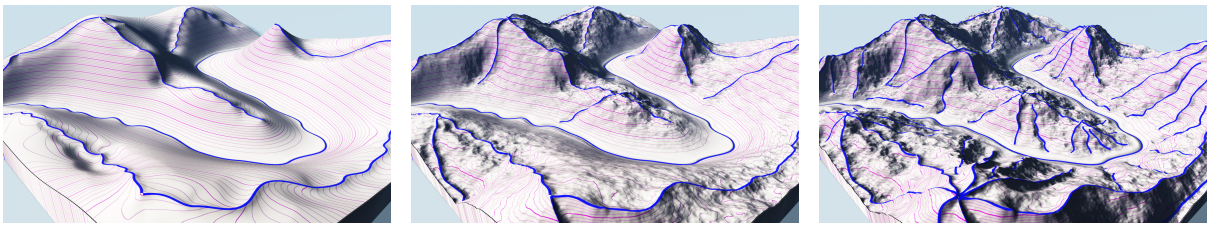


FIG. 64 – édition incrémentale d'un canyon avec 15, 30 et 74 courbes caractéristiques.

FIG. 65 – Ile volcanique (gauche). courbes caractéristique associées (droite).

4.3.2 Contrôle

Un aspect très intéressant et fort de notre approche est sa simplicité et son contrôle. Notre modèle vectoriel fournit à l'utilisateur une représentation compacte et indépendante de la résolution qui peut être éditée et manipulée facilement. Le tableau 1 rapporte des statistiques correspondant aux terrains montrés tout au long de ce chapitre.

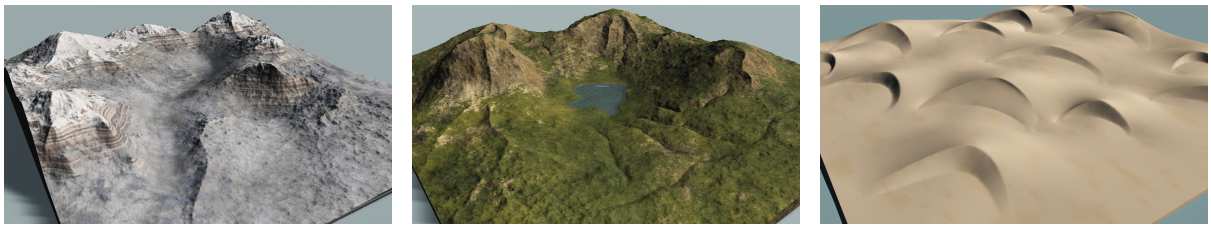


FIG. 66 – Scènes finales de montagne, de lac et de désert rendus avec des textures procédurales.

Une autre caractéristique intéressante de notre méthode est qu'elle fournit un système transparent qui comble la lacune entre l'esquisse et l'édition précise pour la création de terrains complexes. La figure 64 illustre cette capacité avec trois différentes étapes de la création d'un canyon. La première esquisse a été créée en moins de 3 minutes. Il a fallu près de 45 minutes pour modifier attentivement tous les détails nécessaires pour produire le modèle final. Dans ce cas particulier, la majeure partie du temps a été consacrée à ajuster les paramètres de bruit de manière à adapter la rugosité du terrain à la pente. Une amélioration directe pour accélérer la conception de la scène consisterait à calculer directement les paramètres d'amplitude et de rugosité à partir des contraintes de pente.

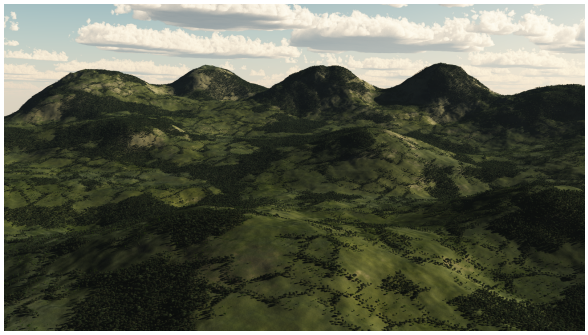


FIG. 67 – Une grande scène avec des collines lisses générées avec des courbes de contrôle dont les gradients sont nuls.

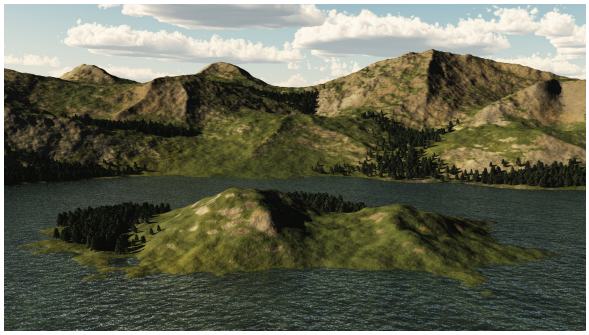


FIG. 68 – Un terrain complexe généré avec notre méthode

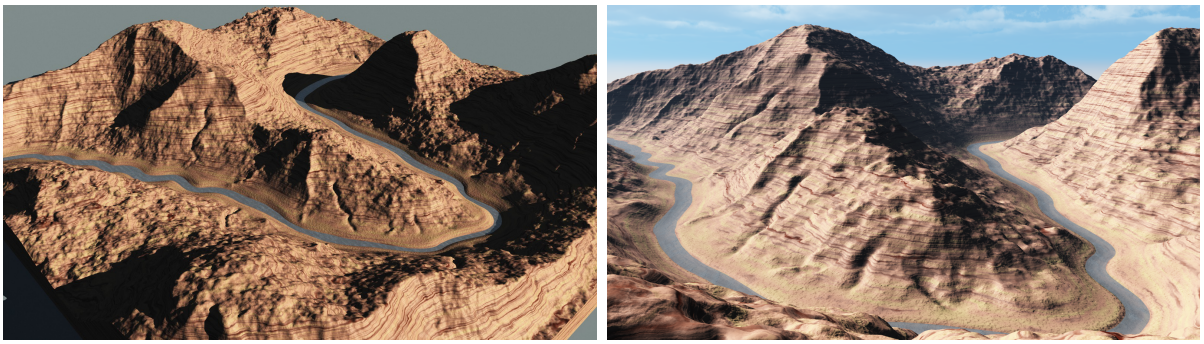


FIG. 69 – Un terrain complexe généré avec notre méthode

4.3.3 Efficacité

Nous avons implémenté nos algorithmes comme des *shaders* de GLSL sur une carte graphique Nvidia 8800 GTS. Le tableau 2 rapporte les temps de calcul pour générer les terrains à différentes résolutions. Les figures montrent que le nombre de courbes caractéristiques a une faible influence sur le temps de génération de terrains, tandis que la taille de la grille a une forte influence sur le temps de génération. Les temps de calcul montrent que notre méthode fonctionne en temps interactif, même pour les grilles de grande taille.

4.4 Conclusion

Dans ce chapitre, nous avons introduit une nouvelle méthode basée sur la diffusion de contraintes en utilisant la méthode de multigrille pour la génération procédurale des terrains réalistes. Notre approche fournit à l'utilisateur des outils simples, intuitifs et efficaces pour contrôler les reliefs du terrain avec un nombre réduit de courbes caractéristiques. Les courbes caractéristiques peuvent remarquablement manipuler la définition de lignes de crête ainsi que les lits de rivières, des lacs et bords de mer d'une façon constante et générique. En outre, notre méthode peut aussi générer des collines et des vallées très facilement. Notre implémentation optimisée sur GPU permet de générer des terrains complexes en temps interactif, même à grandes résolutions (2048×2048). Notre technique peut être utilisée à la fois pour l'esquisse et l'édition en temps réel des modèles très détaillés de terrain.

Conclusion

Dans cette thèse, nous avons proposé plusieurs pistes pour créer des objets naturels avec la possibilité de contrôler localement leur forme. La problématique principale qui est traitée dans notre travail est le contrôle de la génération des objets naturels. À cette fin, nous avons introduit des modèles pour contrôler la génération d'une variété d'objets naturels ainsi que des terrains en utilisant plusieurs techniques de modélisation et d'édition. Chacun de ces modèles présente des avantages et des inconvénients que nous allons détailler.

Notre premier modèle (Chapitre 2) est un modèle multirésolution basé sur les IFS (*Iterated Function Systems*) et équipé d'une notion de détail comme celle de la transformée en ondelettes. Ce modèle présente un outil très efficace pour la synthèse comme pour l'analyse des objets naturels. Il permet aussi d'appliquer des déformations globales sur un objet ainsi que de reconstruire un objet avec peu d'informations. Le besoin d'une étape d'optimisation qui est très coûteuse en temps de calcul, présente un inconvénient majeur de ce modèle. De plus, le contrôle local présenté par ce modèle reste difficile à gérer.

En considérant les avantages et les inconvénients présentés par ce modèle, nous avons proposé un autre modèle multirésolution (Chapitre 3) basé sur la surface de subdivision et équipé aussi d'une notion de détail. L'avantage de ce modèle par rapport aux autres modèles de subdivision est sa capacité de produire des formes rugueuses en plus des formes lisses grâce aux masques fractals utilisés. Nous avons utilisé ce modèle pour générer simplement une variété d'un objet à partir d'un exemple réel (une feuille) tout en choisissant deux ensembles de positions de points de contrôle dans un niveau de résolution, puis nous créons le nombre désiré d'objets entre les deux objets. Nous avons aussi utilisé ce modèle pour générer des terrains en 2.5D et 3D. L'aspect multirésolution de ce modèle permet de créer des caractéristiques de terrain avec des tailles différentes en changeant seulement le niveau de résolution dans lequel la caractéristique est modélisée. Le grand inconvénient de ce modèle est la répétition des détails générés. De plus, le choix des masques n'est pas intuitif.

Dans le chapitre 4, nous nous sommes concentrés sur le problème de la génération de

terrain en proposant un modèle de génération procédurale de terrain à partir d'un ensemble de courbes représentant les caractéristiques de ce terrain. Ces courbes sont appelées courbes caractéristiques. Ce modèle présente plusieurs avantages. Nos courbes caractéristiques sont des primitives génériques pouvant représenter de nombreuses caractéristiques réelles (montagne, lac, colline, rivière, *etc*) et sont aussi des graphiques vectoriels, c'est-à-dire qu'elles sont indépendantes de la résolution du terrain ce qui nous permet d'utiliser le même modèle pour générer un terrain dans différentes résolutions. La taille réduite de la structure de données représentant le modèle et le fait que notre générateur de terrain est un générateur déterministe ce qui permet de générer le même terrain avec le même ensemble de courbes caractéristiques sont des avantages supplémentaires de notre modèle. De plus, la génération est accomplie interactivement grâce à une implémentation sur GPU[5]. Le contrôle présenté par notre modèle va de l'édition de grandes caractéristiques comme les montagnes au contrôle des petits détails ce qui rend le terrain plus réaliste. Les grandes caractéristiques sont gérées par les contraintes de hauteur et de gradient attachées à chaque courbe tandis que les petits détails sont contrôlés par les contraintes de bruit attachées elles aussi à chaque courbe.

La problématique centrale de cette thèse est le contrôle de la forme d'objets naturels. Dans notre travail nous n'avons traité que peu de formes naturelles. Nous pensons que de nombreux objets naturels ne sont pas encore bien étudiés. Pour cette raison le problème abordé dans cette thèse reste très ouvert. Dans la suite nous présentons nos perspectives en présentant les améliorations qui peuvent être développées.

Notre modèle qui est basé sur la subdivision (Chapitre 3) peut traiter plusieurs types d'objets existant dans la nature et surtout les objets rugueux. Nous avons montré que ce modèle est capable de créer les caractéristiques de terrain ainsi que générer une variété de feuilles. Pour la création de terrain, nous avons utilisé un seul ensemble de masques de contrôle, une amélioration serait d'utiliser des ensembles différents de masques dépendant des régions traitées dans le terrain ce qui nous donnerait la possibilité de générer des terrain avec différentes dimensions fractales. Nous avons aussi introduit le principe de la subdivision fractale de maillages arbitraires sans introduire des exemples de l'utilisation de tels modèles. Nous pensons que la subdivision de tels maillages peut être utilisée afin de générer des cailloux par exemple. Une amélioration de ce modèle serait aussi d'intégrer la notion de détails comme celle de la transformée en ondelettes car ces détails enrichissent le modèle et le rendent capable de générer une variété d'objets.

Notre modèle de génération de terrains basée sur des caractéristiques (Chapitre 4) présente un outil très efficace pour contrôler les caractéristiques et même les petits détails

[5]Graphic Processing Unit

de terrain. Ce modèle ouvre plusieurs perspectives à nos yeux. Une amélioration directe serait d'attaquer le problème inverse. Étant donné un terrain représenté par un champ de hauteur, le but serait de trouver les courbes caractéristiques correspondant à ce terrain. Une fois le terrain représenté par des courbes caractéristiques, nous pourrions le reproduire en différentes résolutions grâce à la représentation vectorielle des courbes caractéristiques. Une autre amélioration grâce à cette représentation vectorielle serait de nous permettre de créer les courbes caractéristiques indépendamment du terrain. Cette propriété nous donne la possibilité de créer une base de données de caractéristiques (montagne, rivière, colline, lac, *etc*). Chacune de ces caractéristiques serait utilisée comme un seul objet (une caractéristique possède une ou plusieurs courbes avec des points de contrainte). La méthode de diffusion garantit le mélange sans effets artificiels des caractéristiques sur le terrain. La création de base de données de caractéristiques accélère la génération de terrain en diminuant le temps nécessaire pour modéliser les courbes. La visualisation multirésolution dépendant du point de vue serait un prolongement intéressant de notre modèle vectoriel. Celle-ci est possible grâce à la représentation vectorielle des courbes caractéristiques et l'utilisation de GPU[5]. La reconstruction de terrain à partir de données incomplètes peut aussi être un prolongement intéressant de notre modelé de diffusion. L'idée est de prendre une petit nombre d'échantillons d'un terrain réel et d'utiliser cet échantillonnage pour reconstruire le terrain original sans passer par les courbes caractéristiques. Nous pensons aussi que nous pouvons prolonger notre modèle afin de permettre de modéliser des caractéristiques artificielles comme les routes par exemple.

Bibliographie

[APS09] Fabricio Anastacio, Przemyslaw Prusinkiewicz, and Mario Costa Sousa. Sketch-based parameterization of l-systems using illustration-inspired construction lines and depth modulation. *Computers & Graphics*, 33(4):440–451, 2009.

[BA05a] Farès Belhadj and Pierre Audibert. Modeling landscapes with ridges and rivers. In *VRST*, pages 151–154, 2005.

[BA05b] Farès Belhadj and Pierre Audibert. Modeling landscapes with ridges and rivers: bottom up approach. In *GRAPHITE*, pages 447–450, 2005.

[Bar88] Michael Barnsley. *Fractals everywhere.* Academic Press, 1988.

[Bar93] Michael Barnsley. *Fractals everywhere (second edition).* Academic Press Professional, 1993.

[BDHJ04] Martin Bertram, Mark A. Duchaineau, Bernd Hamann, and Kenneth I. Joy. Generalized b-spline subdivision-surface wavelets for geometry compression. *IEEE Transactions on Visualization and Computer Graphics*, 10(3):326–338, 2004.

[Bel07] Farès Belhadj. Terrain modeling: a constrained fractal model. In *Afrigraph*, pages 197–204, 2007.

[Ben82] Mandelbrot Benoit. *The Fractal Geometry of Nature.* Freeman, San Fransisco, 1982.

[BFGS03] Jeffrey Bolz, Ian Farmer, Eitan Grinspun, and Peter Schröder. Sparse matrix solvers on the gpu: conjugate gradients and multigrid. *ACM Trans. Graph.*, 22(3):917–924, 2003.

[BMBZ02] Henning Biermann, Ioana M. Martin, Fausto Bernardini, and Denis Zorin. Cut-and-paste editing of multiresolution surfaces. *ACM Trans. Graph.*, 21(3):312–321, 2002.

[BPF+03] F. Boudon, P. Prusinkiewicz, P. Federl, C. Godin, and R. Karwowski. Interactive design of bonsai tree models. *Computer Graphics Forum. Proceedings of Eurographics*, 22(3):591–599, 2003.

[BSS06] John Brosz, Faramarz F. Samavati, and Mario Costa Sousa. Terrain synthesis by-example. In José Braz, Joaquim A. Jorge, Miguel Dias, and Adérito Marcos, editors, *GRAPP*, pages 122–133. INSTICC - Institute for Systems and Technologies of Information, Control and Communication, 2006.

[CC78] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350–355, Septemper 1978.

[CD05] Robert L. Cook and Tony DeRose. Wavelet noise. *ACM Trans. Graph.*, 24(3):803–811, 2005.

[Cha74] G. Chaikin. An algorithm for high speed curve generation. In *Computer Graphics and Image Processing*, volume 3, pages 346–349, 1974.

[CHZ00] Jonathan M. Cohen, John F. Hughes, and Robert C. Zeleznik. Harold: aworld made of drawings. In *NPAR*, pages 83–90, 2000.

[CMN98] Jianyun Chai, Takaharu Miyoshi, and Eihachiro Nakamae. Contour interpolation and surface reconstruction of smooth terrain models. In *IEEE Visualization*, pages 27–33, 1998.

[CNX+08] Xuejin Chen, Boris Neubert, Ying-Qing Xu, Oliver Deussen, and Sing Bing Kang. Sketch-based tree modeling using markov random field. *ACM Trans. Graph.*, 27(5):109, 2008.

[Coo84] Robert L Cook. Shade trees. *Computer Graphics*, vol. 18(no. 3):pp: 223–230, July 1984.

[CZ09] Yang Chongjun Chen Zhuo, Zhao Yanqing. Real-time contour map reconstruction with 3d terrain on modern graphics hardware. In *international workshop on Virtual Changing Globe for Visualisation and Analysis (VCGVA2009)*, Wuhan, China, October 2009.

[Dac06] Carsten Dachsbacher. *Interactive Terrain Rendering:Towards Realism with Procedural Models and Graphics Hardware*. Phd thesis, Universität Erlangen-Nürnberg zur Erlangung des Grades, 2006.

[Dei03] Wolfgang Deix. Real-time rendering of fractal rocks. In *The 7th Central European Seminar on Computer Graphics*, Slovakia, April 2003.

[DL97] Oliver Deussen and Bernd Lintermann. A modelling method and user interface for creating plants. In Wayne A. Davis, Marilyn M. Mantei, and R. Victor Klassen, editors, *Graphics Interface*, pages 189–198. Canadian Human-Computer Communications Society, 1997.

[dREF+88] Philippe de Reffye, Claude Edelin, Jean Françon, Marc Jaeger, and Claude Puech. Plant models faithful to botanical structure and developmentr. In Richard J. Beach, editor, *SIGGRAPH*, pages 151–158. ACM, 1988.

[DS78] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design*, 10(6), 1978.

[EBL07] Sharon E., Roman B., and Swinney H. L. Geometrically driven wrinkling observed in free plastic sheets and leaves. *Physical Review E*, 75(4):046211, apr 2007.

[Edg90] Gerald A. Edgar. *Measure, Topology, and Fractal Geometry*. Springer Verlag, 1990.

[EG01] James H. Elder and Richard M. Goldberg. Image editing in the contour domain. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(3):291–296, 2001.

[Elb01] Gershon Elber. Multiresolution curve editing with linear constraints. *J. Comput. Inf. Sci. Eng.*, 1(4):347–355, 2001.

[FB88] David R. Forsey and Richard H. Bartels. Hierarchical b-spline refinement. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 205–212, New York, NY, USA, 1988. ACM.

[FFC82a] Alain Fournier, Don Fussell, and Loren Carpenter. Computer rendering of stochastic models. *Commun. ACM*, 25(6):371–384, 1982.

[FFC82b] Alain Fournier, Donald S. Fussell, and Loren C. Carpenter. Computer rendering of stochastic models. *Commun. ACM*, 25(6):371–384, 1982.

[FLW02] Raanan Fattal, Dani Lischinski, and Michael Werman. Gradient domain high dynamic range compression. In *SIGGRAPH*, pages 249–256, 2002.

[FS94] Adam Finkelstein and David Salesin. Multiresolution curves. In *SIGGRAPH*, pages 261–268, 1994.

[Gar85] Geoffrey Y. Gardner. Visual simulation of clouds. In *SIGGRAPH*, pages 297–304, 1985.

[Gar88] Geoffrey Y. Gardner. Siggraph course notes: Functional modelling. In *SIGGRAPH*, Atlanta, 1988.

[GM01] M. Gamito and F. K. Musgrave. Procedural landscapes with overhangs. In *In 10th Portuguese Computer Graphics Meeting*, pages 33–42, 2001.

[GMS09] James Gain, Patrick Marais, and Wolfgang Strasser. Terrain sketching. In *I3D '09: Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 31–38, New York, NY, USA, 2009. ACM.

[G.S88] Miller G.S.P. personal communications, 1988.

[GTB02] Eric Guérin, Eric Tosan, and Atilla Baskurt. Modeling and approximation of fractal surfaces with projected IFS attractors. In M. M. Novak, editor, *Emergent Nature, Fractal 2002 proceedings*, pages 293–303. World Scientific, mar 2002.

[Gué02] Eric Guérin. *Approximation fractale de courbes et de surfaces.* Thèse de doctorat, Université Claude Bernard Lyon 1, dec 2002.

[GWL+03] Nolan Goodnight, Cliff Woolley, Gregory Lewin, David P. Luebke, and Greg Humphreys. A multigrid solver for boundary value problems using programmable graphics hardware. In *Graphics Hardware*, pages 102–111, 2003.

[Han07] Jinshu Han. Plant simulation based on fusion of l-system and ifs. In *International Conference on Computational Science (2)*, pages 1091–1098, 2007.

[HKD93] Mark Halstead, Michael Kass, and Tony DeRose. Efficient, fair interpolation using catmull-clark surfaces. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 35–44, New York, NY, USA, 1993. ACM.

[HSB05] Sung Min Hong, R. Bruce Simpson, and Gladimir V. G. Baranoski. Interactive venation-based leaf shape modeling. *Journal of Visualization and Computer Animation*, 16(3-4):415–427, 2005.

[HSS03] Kai Hormann, Salvatore Spinello, and Peter Schröder. C1-continuous terrain reconstruction from sparse contours. In Thomas Ertl, editor, *VMV*, pages 289–297. Aka GmbH, 2003.

[Hut81] John E. Hutchinson. Fractals and self-similarity. *Indiana University Journal of Mathematics*, 30:713–747, 1981.

[IOI06a] Takashi Ijiri, Shigeru Owada, and Takeo Igarashi. Seamless integration of initial sketching and subsequent detail editing in flower modeling. *Comput. Graph. Forum*, 25(3):617–624, 2006.

[IOI06b] Takashi Ijiri, Shigeru Owada, and Takeo Igarashi. The sketch l-system: Global control of tree modeling using free-form strokes. In Andreas Butz, Brian D. Fisher, Antonio Krüger, and Patrick Olivier, editors, *Smart Graphics*, volume 4073 of *Lecture Notes in Computer Science*, pages 138–146. Springer, 2006.

[IOOI05] Takashi Ijiri, Shigeru Owada, Makoto Okabe, and Takeo Igarashi. Floral diagrams and inflorescences: interactive flower modeling using botanical structural constraints. *ACM Trans. Graph.*, 24(3):720–726, 2005.

[Jur89] Dietmar Saupe Jurgens, H. Point evaluation of multi-variable random fractals. *Springer-Verlag Heidelberg*, 1989.

[KS99] Andrei Khodakovsky and Peter Schröder. Fine level feature editing for subdivision surfaces. In *Symposium on Solid Modeling and Applications*, pages 203–211, 1999.

[Lew87a] John P. Lewis. Generalized stochastic subdivision. *ACM Trans. Graph.*, 6(3):167–190, 1987.

[Lew87b] John P. Lewis. Generalized stochastic subdivision. *ACM Trans. Graph.*, 6(3):167–190, 1987.

[Lin68] Aristid Lindenmayer. Mathematical models for cellular interactions in development ii. simple and branching filaments with two-sided inputs. *Journal of Theoretical Biology*, 18(3):300–315, March 1968.

[Loo87] C. Loop. Smooth subdivision surfaces based on triangles. Department of mathematics, University of Utah, Utah, USA, Aug 1987.

[Mal89] Stéphane Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(7):674–693, 1989.

[Man77] Benoit Mandelbrot. *Fractal Geometry of Nature*. W. H. Freeman, 1977.

[Man88] Benoit B. Mandelbrot. Fractal landscapes without creases and with rivers. pages 243–260, 1988.

[Mey87] Y. Meyer. Les ondelettes. In *Contributions to nonlinear partial differential equations, Vol. II (Paris, 1985)*, volume 155 of *Pitman Res. Notes Math. Ser.*, pages 158–171. Longman Sci. Tech., Harlow, 1987.

[Mil86] Gavin S. P. Miller. The definition and rendering of terrain maps. In *SIGGRAPH*, pages 39–48, 1986.

[MJW+05] Chiang. M., Huang. J., Tai. W., Liu. C., and Chang. C. Terrain synthesis: An interactive approach. In *International Workshop on Advanced Image Tech (2005)*, 2005.

[MKM89] F. Kenton Musgrave, Craig E. Kolb, and Robert S. Mace. The synthesis and rendering of eroded fractal terrains. In *SIGGRAPH*, pages 41–50, 1989.

[MM01] Marryat Ma and Stephen Mann. Multiresolution editing of pasted surfaces. In *Mathematical Methods for Curves and Surfaces: Oslo 2000*, pages 273–282, Nashville, TN, USA, 2001. Vanderbilt University.

[MMD04] Alex Reche Martinez, Ignacio Martín, and George Drettakis. Volumetric reconstruction and interactive rendering of trees from photographs. *ACM Trans. Graph.*, 23(3):720–727, 2004.

[MMPP03] Lars Mündermann, Peter MacMurchy, Juraj Pivovarov, and Przemyslaw Prusinkiewicz. Modeling lobed leaves. In *Computer Graphics International*, pages 60–67, 2003.

[MP96] Radomír Mech and Przemyslaw Prusinkiewicz. Visual models of plants interacting with their environment. In *SIGGRAPH*, pages 397–410, 1996.

[MP08] James McCann and Nancy S. Pollard. Real-time gradient-domain painting. *ACM Trans. Graph.*, 27(3), 2008.

[MVN68] Benoit B. Mandelbrot and John W. Van Ness. Fractional brownian motions, fractional noises and applications. *SIAM Review*, 10(4):422–437, 1968.

[NFD07] Boris Neubert, Thomas Franken, and Oliver Deussen. Approximate image-based tree-modeling using particle flows. *ACM Trans. Graph.*, 26(3):88, 2007.

[OBW+08] Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. Diffusion curves: a vector representation for smooth-shaded images. *ACM Trans. Graph.*, 27(3), 2008.

[OOI05] Makoto Okabe, Shigeru Owada, and Takeo Igarashi. Interactive design of botanical trees using freehand sketches and example-based editing. *Comput. Graph. Forum*, 24(3):487–496, 2005.

[Per85] Ken Perlin. An image synthesizer. In *SIGGRAPH*, pages 287–296, 1985.

[PFTV93a] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C : The Art of Scientific Computing*, chapter Nonlinear Models. Cambridge University Press, 1993.

[PFTV93b] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C : The Art of Scientific Computing*, chapter Partial Differential Equations. Cambridge University Press, 1993.

[PGB03] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318, 2003.

[PGGM09] Adrien Peytavie, Eric Galin, Jérôme Grosjean, and Stéphane Mérillou. Procedural generation of rock piles using aperiodic tiling. *Comput. Graph. Forum*, 28(7):1801–1809, 2009.

[PGMG09] Adrien Peytavie, Eric Galin, Stephane Merillou, and Jerome Grosjean. Arches: a Framework for Modeling Complex Terrains. *Computer Graphics Forum (Proceedings of Eurographics)*, 28(2):457–467, 2009.

[PGTG04] Joachim Pouderoux, Jean-Christophe Gonzato, Ireneusz Tobor, and Pascal Guitton. Adaptive hierarchical rbf interpolation for creating smooth digital elevation models. In Dieter Pfoser, Isabel F. Cruz, and Marc Ronthaler, editors, *GIS*, pages 232–240. ACM, 2004.

[PHL+09] Wojciech Palubicki, Kipp Horel, Steven Longay, Adam Runions, Brendan Lane, Radomír Mech, and Przemyslaw Prusinkiewicz. Self-organizing tree models for image synthesis. *ACM Trans. Graph.*, 28(3), 2009.

[PHM93] Przemyslaw Prusinkiewicz, Mark Hammel, and Eric Mjolsness. Animation of plant development. In *SIGGRAPH*, pages 351–360, 1993.

[PJM94] Przemyslaw Prusinkiewicz, Mark James, and Radomír Mech. Synthetic topiary. In *SIGGRAPH*, pages 351–358, 1994.

[PL91] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants (The Virtual Laboratory)*. Springer, October 1991.

[PMKL01] Przemyslaw Prusinkiewicz, Lars Mündermann, Radoslaw Karwowski, and Brendan Lane. The use of positional information in the modeling of plants. In *SIGGRAPH*, pages 289–300, 2001.

[PP93] HAMMEL M. PRUSINKIEWICZ P. A fractal model of mountains with rivers. *Proceeding of Graphics Interface*, pages 174–180, May 1993.

[QTZ+06] Long Quan, Ping Tan, Gang Zeng, Lu Yuan, Jingdong Wang, and Sing Bing Kang. Image-based plant modeling. *ACM Trans. Graph.*, 25(3):599–604, 2006.

[RCSL04] Yodthong Rodkaew, Prabhas Chongstitvatana, Suchada Siripant, and Chidchanok Lursinsap. Modeling plant leaves in marble-patterned colours with particle transportation system. In *4th International Workshop on Functional-Structural Plant Models*, pages 391–397, Montpellier, France, 2004.

[RdlF05] Abigail Martínez Rivas and Luis Gerardo de la Fraga. Terrain reconstruction from contour maps. In *Proceedings of the 14th International Congress on Computing (CIC2005)*, pages 167–175, Mexico City, September 2005.

[RFL+05] Adam Runions, Martin Fuhrer, Brendan Lane, Pavol Federl, Anne-Gaëlle Rolland-Lagan, and Przemyslaw Prusinkiewicz. Modeling and visualization of leaf venation patterns. *ACM Trans. Graph.*, 24(3):702–711, 2005.

[RLFS02] Yodthong Rodkaew, Chidchanok Lursinsap, Tadahiro Fujimoto, and Suchada Siripant. Modeling leaf shapes using l-systems and genetic algorithms. In *In International Conference NICOGRAPH (April*, pages 73–78, 2002.

[RLP07] Adam Runions, Brendan Lane, and Przemyslaw Prusinkiewicz. Modeling Trees with a Space Colonization Algorithm. pages 63–70, Prague, Czech Republic, 2007. Eurographics Association.

[RME09] Brennan Rusnell, David Mould, and Mark G. Eramian. Feature-rich distance-based terrain synthesis. *The Visual Computer*, 25(5-7):573–579, 2009.

[Sau88] Dietmar Saupe. Algorithms for random fractals. pages 71–113, 1988.

[SCX09] Lu Shenglian, Zhao Chunjiang, and Guo Xinyu. Venation skeleton-based modeling plant leaf wilting. *Int. J. Comput. Games Technol.*, 2009:1–8, 2009.

[SFS05] Lisa Streit, Pavol Federl, and Mario Costa Sousa. Modelling plant variation through growth. *Comput. Graph. Forum*, 24(3):497–506, 2005.

[SJ04] Byeong-Seok Shin and Hoe Sang Jung. Fast reconstruction of 3d terrain model from contour lines on 2d maps. In Doo-Kwon Baik, editor, *AsiaSim*, volume 3398 of *Lecture Notes in Computer Science*, pages 230–239. Springer, 2004.

[SJ05] Byeong-Seok Shin and Hoe Sang Jung. Contour-based terrain model reconstruction using distance information. In Osvaldo Gervasi, Marina L. Gavrilova, Vipin Kumar, Antonio Laganà, Heow Pueh Lee, Youngsong Mun, David Taniar, and Chih Jeng Kenneth Tan, editors, *ICCSA (3)*, volume 3482 of *Lecture Notes in Computer Science*, pages 1177–1186. Springer, 2005.

[SLSS06] L. Streit, P. Lapides, M. Costa Sousa, and E. Sharlin. Modeling plant variations through 3d interactive sketches. In *in 3rd Eurographics Workshop on Sketch-Based Interfaces and Modeling (SBIM 2006)*, Vienna, Austria, September 2006.

[SRDT01] Ilya Shlyakhter, Max Rozenoer, Julie Dorsey, and Seth J. Teller. Reconstructing 3d tree models from instrumented photographs. *IEEE Computer Graphics and Applications*, 21(3):53–61, 2001.

[SS05] Szymon Stachniak and Wolfgang Stuerzlinger. An algorithm for automated fractal terrain deformation. In *Computer Graphics and Artificial Intelligence*, pages 64–76, May 2005.

[TBSS+07] E. Tosan, I. Bailly-Salins, I. Stotz, G. Gouaty, and Y. Weinand. Modelisation iterative de courbes et surfaces : aspect multiresolution. In *Groupe de travail en Modelisation Geometrique journee de Valenciennes*, pages 55–69, mars 2007.

[TG00] David Thibault and Christopher M. Gold. Terrain reconstruction from contours by skeleton construction. *GeoInformatica*, 4(4):349–373, 2000.

[THT08] Shih-Chun Tu, Chun-Yen Huang, and Wen-Kai Tai. Terrain synthesis based on microscopic terrain feature. In Zhigeng Pan, Xiaopeng Zhang, Abdennour El Rhalibi, Woontack Woo, and Yi Li, editors, *Edutainment*, volume 5093 of *Lecture Notes in Computer Science*, pages 644–655. Springer, 2008.

[TZW+07] Ping Tan, Gang Zeng, Jingdong Wang, Sing Bing Kang, and Long Quan. Image-based tree modeling. *ACM Trans. Graph.*, 26(3):87, 2007.

[UWRE03] Nath Utpal, Crawford Brian C. W., Carpenter Rosemary, and Coen Enrico. Genetic control of surface curvature. *Science*, 299(5611):1404–1407, February 2003.

[Vos85] Richard F Voss. Random fractal forgeries. *Springer-Verlag, Berlin*, 1985. In Fundamental Algorithms for Computer Graphics, R. A. Earnshaw, ed.

[WBCG09] Jamie Wither, Frédéric Boudon, Marie-Paule Cani, and Christophe Godin. Structure from silhouettes: a new paradigm for fast sketch-based design of trees. *Comput. Graph. Forum*, 28(2):541–550, 2009.

[WI04] Nayuko Watanabe and Takeo Igarashi. A sketching interface for terrain modeling. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Posters*, page 73, New York, NY, USA, 2004. ACM.

[WP95] Jason Weber and Joseph Penn. Creation and rendering of realistic trees. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 119–128, New York, NY, USA, 1995. ACM.

[WV03] John C. Whelan and Mahes Visvalingam. Formulated silhouettes for sketching terrain. In *TPCG*, pages 90–96. IEEE Computer Society, 2003.

[WY04] Qing Wu and Yizhou Yu. Feature matching and deformation for texture synthesis. *ACM Trans. Graph.*, 23(3):364–367, 2004.

[Zai98] Chems Eddine Zair. *Formes fractales à pôles basées sur une généralisation des IFS*. Thèse de doctorat, Université Claude Bernard Lyon 1, jun 1998.

[ZSTR07] Howard Zhou, Jie Sun, Greg Turk, and James M. Rehg. Terrain synthesis from digital elevation models. *IEEE Trans. Vis. Comput. Graph.*, 13(4):834–848, 2007.

[ZT96] Chems Eddine Zaïr and Eric Tosan. Fractal modeling using free form techniques. *Comput. Graph. Forum*, 15(3):269–278, 1996.

Publications

Publications internationales

Hnaidi Houssam, Gurin ric, Akkouche Samir, Peytavi Adrien, Galin ric. Feature based terrain generation using diffusion equation. Pacific Graphic. To appear. September 2010.

Hnaidi Houssam, Gurin ric, Akkouche Samir. Multiresolution control of curves and surfaces with a self-similar model. Fractals Journal. Vol. 18 No. 3. September 2010.

Hnaidi Houssam, Gurin ric, Akkouche Samir. Fractal/Wavelet representation of objects. In the international conference on information & communication technologies : from theory to application ICTTA'08. April 2008.

Publications nationales

Hnaidi Houssam, Gurin ric, Akkouche Samir. Insertion de dtail dans des figures autosimilaires. Dans AFIG. Novembre 2007.

Résumé

La génération de formes naturelles a été le sujet de nombreuses recherches depuis plusieurs années. Plusieurs méthodes ont été proposées afin de générer des objets naturels et réalistes tels que des terrains, des plantes et arbres, des nuages, etc. Les modèles itératifs sont très connus dans ce domaine de recherche grâce à leur capacité à générer des formes rugueuses et complexes qui sont adaptées à la représentation d'objets naturels. L'inconvénient majeur de tels modèles est le manque de contrôle sur le résultat final. Ce dernier peut venir de la méthode de construction stochastique interdisant tout contrôle par définition. Pour les modèles dont la construction est déterministe, les paramètres de générations sont souvent non intuitifs et limitent ainsi le contrôle. Pour ces raisons un grand nombre de recherches ont port sur le problème du contrôle de ces modèles ainsi que sur la possibilité d'utiliser des modèles non-itératifs (esquisses, basés exemples, etc.). Bien souvent, le contrôle introduit par ces modèles est un contrôle global, c'est-à-dire sur la totalité de l'objet final et ne prend donc pas en compte les détails locaux de ce dernier.

Dans notre travail, nous nous attaquons au problème du contrôle sur les formes naturelles en tenant compte du contrôle local. À cette fin, nous introduisons deux modèles différents. Le premier repose sur un formalisme itératif avec notion de détail qui se dicline en deux sous-familles, l'une basée sur les IFS et l'autre basée sur les surfaces de subdivision. Le deuxième modèle permet l'édition de caractéristiques d'un terrain sous forme de primitives vectorielles puis la génération du terrain par une méthode de diffusion guidée. Cette dernière fait l'objet d'une implémentation parallèle sur la carte graphique (GPU).

Mots-clés: Fractal, Mulirésolution, Modélisation de terrain, Génération de terrain, Génération procédurale, diffusion, Graphiques vectoriels.

Abstract

The generation of natural shapes has been the subject of much research for many years. Several methods have been proposed to generate realistic natural objects such as terrain, plants and trees, clouds, etc... Iterative models are well known in this field of research due to their ability to generate complex and rough shapes that are adapted to the representation of natural objects. The major drawback of such models is the lack of control over the final result. The latter can come from the stochastic construction method which prevents any control by definition. For models whose construction is deterministic, the parameters of generation are often non-intuitive and thus limit control. For these reasons many studies have focused on the problem of controlling these models as well as the possibility of using non-iterative models (sketches, based on examples, *etc*). Often, control introduced by these models is a global control, on the whole final object and therefore does not include local details of this object.

In our work, we focus in the problem of control over natural shapes, taking into account local control. To this end, we introduce two different models. The first is based on an iterative formalism with detail concept which is divided into two subfamilies, one based on IFS and the other one based on subdivision surfaces. The second model allows the editing of terrain features under a form of vectorial primitives which one used to generate the terrain by guided diffusion method. The latter is the subject of a parallel implementation on graphics card (GPU).

Keywords: Fractal, Multiresolution, Terrain modeling, Terrain generation, Procedural generation, diffusion, Vector graphics.