Synthèse Aléatoire de Terrains 3D

Erwan Le Martelot

27 janvier 2005

Table des matières

Résumé

L'objectif de cet article est de présenter quelques techniques pour la synthèse de terrains réalistes, ainsi que leurs limites tant par leur temps de calcul que par leurs possibilités. Aux algorithmes de génération de terrain de base seront combinées ou ajoutées des méthodes d'amélioration de ce terrain pouvant entre autres, faire passer une vallée (ou rivière) parmi une chaîne de montagnes, ou créer des zones planes au milieu des roches, afin de rendre l'ensemble plus proche du réel. Pour cela seront présentées deux approches, tirées des deux articles [1] et [2] : d'une part l'algorithme fractal "squig-curves" et d'autre part le principe d'érosion, inspiré de la physique.

Introduction

La synthèse aléatoire d'environnements réalistes est une notion que l'on retrouve au coeur d'applications comme la simulation ou le jeu. Son principe est de parvenir à créer de manière totalement aléatoire un environnement virtuel mais pour le moins réaliste. Le but est donc de trouver un moyen de créer de tels environnements en approchant au mieux les principes esthétiques de la nature. La découverte des fractals, dans les années 1970, par le mathématicien Benoît Mandelbrot, va ouvrir une voie vers de nouveaux algorithmes permettant de telles créations. Terrains 3D, arbres, fougères, plantes de tous genres, peuvent maintenant être assez facilement créés par des algorithmes fractals.

Nous allons nous intéresser dans cet article à la génération de terrains 3D. L'approche fractale va permettre par des procédés simples et relativement intuitifs de créer des paysages montagneux basiques. Cependant, les possibilités offertes par ces méthodes élémentaires restent très limitées. Les diagrammes de Voronoï permettent de créer des archipels mais sans le réalisme des approches fractales. Par ailleurs, la puissance de calcul requise pour générer de tels environnement est déjà très importante. Il est donc nécessaire de trouver des méthodes qui offrent de plus larges possibilités de génération au moindre coût de calcul. Je vais pour cela me baser principalement sur les articles [1] et [2] présentant des méthodes de génération de terrains, plutôt rocheux, avec le soucis de la contrainte importante de génération "temps réel".

Dans un premier temps, j'expliquerai quelques approches élémentaires de synthèse de terrains que l'on retrouve comme base très fréquemment. Nous aborderons notamment la génération de terrain montagneux avec les algorithmes "Spectral synthesis", "Mid-point displacement" et diagrammes de

Voronoï. Dans un second temps, j'aborderai des méthodes d'amélioration de la génération de ce terrain dont la méthode "Squig-curves" de génération de rivières présentée dans [1]. Enfin, je présenterai des méthodes de retouche du terrain généré dont l'approche originale, inspirée de la physique et présentée dans [2], les méthode par érosion.

A nos approches basiques, toutes ces méthodes ajouterons ou coupleront des techniques d'amélioration des possibilités de génération, ou de qualité du rendu. Elles donneront ainsi un résultat plus proche de ce que l'on peut rencontrer en réalité en minimisant au possible le coût de calcul.

L'article présentera les possibilités et limites de chaque méthode. Chacune possédant avantages et inconvénients, nous aborderons une démarche progressive dans laquelle nous amélioreront pas à pas la synthèse de terrain, selon des choix parmi ces méthodes et des critères les justifiant. Nous verrons en synthèse ce que globalement ces méthodes permettent de faire, et, en étudiant leurs limites, où l'approche globale se voit restreinte.

1 Génération de terrain de base

Dans un premier temps, nous allons voir quelques méthodes de génération de terrains de base. Il en existe un certain nombre, on peut citer les algorithmes :

- Spectral synthesis (Synthèse spectrale),
- Mid-point displacement (Déplacement du point-médian),
- Diagrammes de Voronoï,
- Fault (Défaut),
- Circles.

Tous génèrent des grilles (à base carrée ou triangulaire) et la valeur de chaque case représente la hauteur du point.

Cette partie présentera les trois premiers algorithmes cités, donnant le premier jet à la synthèse de terrains 3D.

1.1 Algorithmes de type fonction à bruit

Ce type d'algorithme est fréquemment utilisé pour la génération de terrain de base. Le principe est de modeler un terrain en appliquant successivement un bruit de plus en plus petit. Deux algorithmes de ce type sont présentés ici : "Mid-point displacement" et "Spectral synthesis".

1.1.1 Algorithme “Mid-point displacement”

Le principe de cet algorithme est de calculer au fur et à mesure les valeurs manquantes, milieux de segments ou centres de carrés dont les extrémités sont connues. La nouvelle valeur est attribuée en effectuant la moyenne des valeurs connues et en ajoutant un bruit. L'algorithme commence avec le terrain entier puis le subdivise et recommence de même sur chaque sous partie. Afin de respecter le principe fractal de base, comme quoi un phénomène se répète identiquement à son échelle dans toute sous partie de l'objet considéré, l'intervalle des valeurs possibles de ce bruit est réduit en même temps que la taille de la sous partie traitée.

Cet algorithme nécessite deux paramètres au départ. Le premier est appelé “détail”, il détermine la taille de la grille donc la quantité de points. Le second, qui détermine l'évolution de l'intervalle des valeurs possibles pour le bruit, est appelé “smoothness” ou “roughness”. Ce paramètre porte ce nom (“douceur” ou “rugosité”) car en fonction de sa valeur, le bruit pourra être fort comme faible, autrement dit les changements de hauteur des points pourront être doux ou brutaux.

Cet algorithme existe sous deux formes :

- base carrée (Diamond-square),
- base triangulaire.

Base carrée

La forme à base carrée fonctionne sur une grille de côté $2^n + 1$. Cette forme s'appelle aussi algorithme “Diamond-Square” car il se fait en deux étapes alternatives, l'étape “diamond” et l'étape “square”. L'étape “diamond” calcule la valeur du centre d'un carré, tandis que l'étape “square” calcule les valeurs des milieux des quatre segments de ce carré (Figure 1).

Phase “Diamond” : Calcul de la valeur du centre du carré. On effectue la moyenne des valeurs des coins du carré puis on ajoute un bruit tempéré par un facteur de déplacement aléatoire appelé ici “disp_factor”, recalculé à chaque itération en fonction de la rugosité. Soient h la fonction donnant la hauteur d'un point p, et $random$ une fonction renvoyant un réel aléatoire compris entre 0 et 1. Sur la phase Diamond 1 :

$$h(p_0) = \frac{h(p_1) + h(p_2) + h(p_3) + h(p_4)}{4} + random() \times disp\_factor$$

Phase “Square” : Calcul de la valeur du milieu de chaque côté du carré précédent. Effectuer la moyenne des points du carré-losange dont chaque point à calculer est le centre, et ajouter de même un bruit. Sur la phase

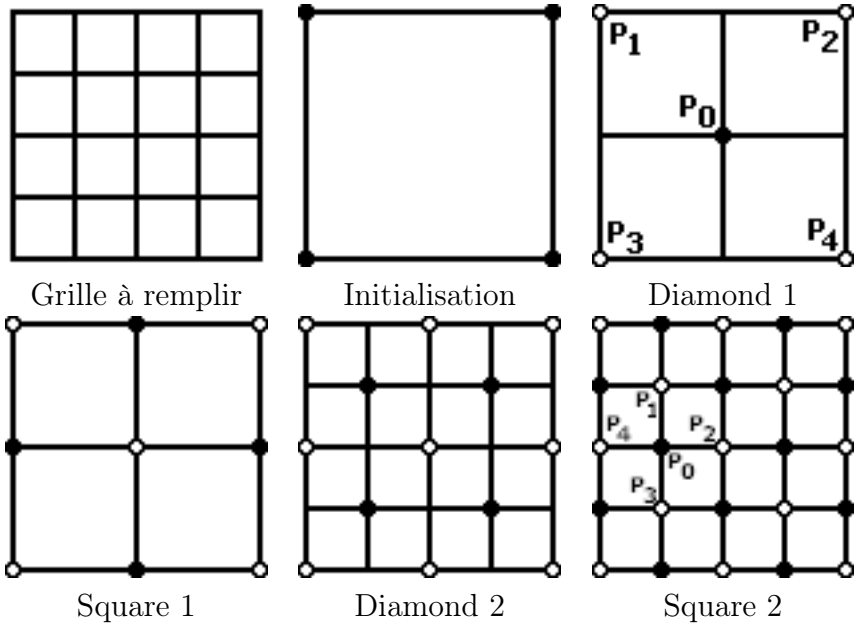


FIG. 1 – Algorithme Mid-point displacement à base carrée (Algorithme Diamond-Square) : Initialement les points de la grille n'ont pas de valeur. Une valeur aléatoire est attribuée aux coins lors de l'initialisation. Ensuite la phase "diamond" attribue une valeur au centre du carré, puis la phase "square" à chaque milieu de segment du carré. Ensuite le processus recommence pour les sous carrés.

Square 2 :

$$h(p_0) = \frac{h(p_1) + h(p_2) + h(p_3) + h(p_4)}{4} + random() \times disp\_factor$$

La figure 2 montre deux terrains générés par cette méthode, avec des rugosités différentes.

Base triangulaire

Cette forme consiste à subdiviser successivement un triangle en quatre triangles. Une valeur est attribuée à chaque milieu de côté de chaque triangle (Figure 3).

Ici, a chaque itération, les trois milieux de segments sont calculés en fonction de la moyenne des valeurs des trois sommets du triangle, à laquelle s'ajoute un bruit tempéré par le facteur de déplacement aléatoire.

$$h(p) = \frac{h(p_1) + h(p_2) + h(p_3)}{3} + random() \times disp\_factor$$

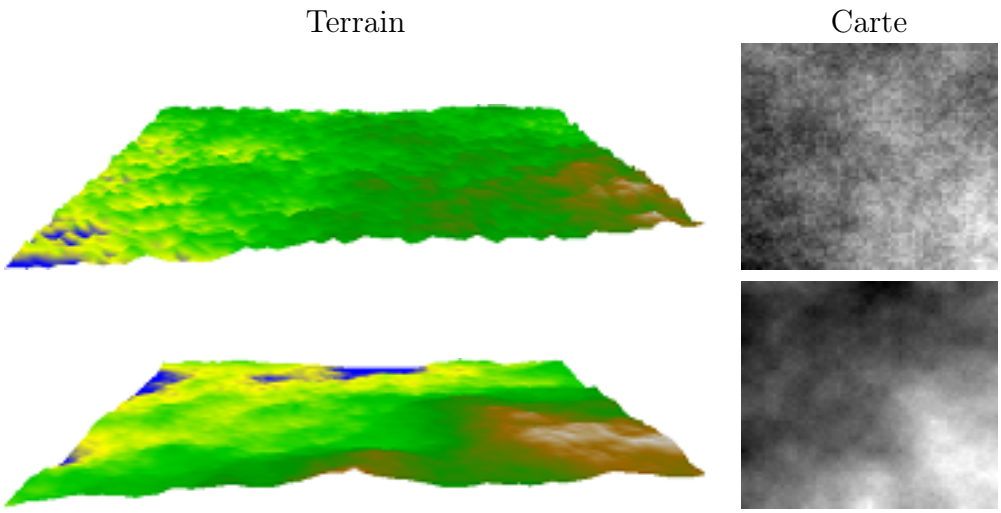


FIG. 2 – Terrain généré par l'algorithme Mid-point displacement à base carrée. Le terrain en haut est généré avec une rugosité de 0.1 et celui du bas avec une rugosité de 0.9. On observe que celui du haut est très irrégulier alors que celui du bas est plus constant en terme de changement de relief.

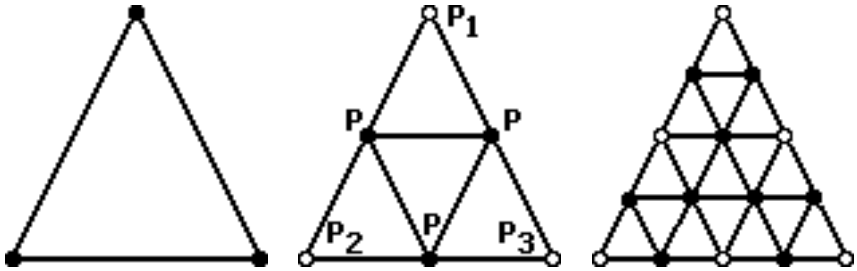


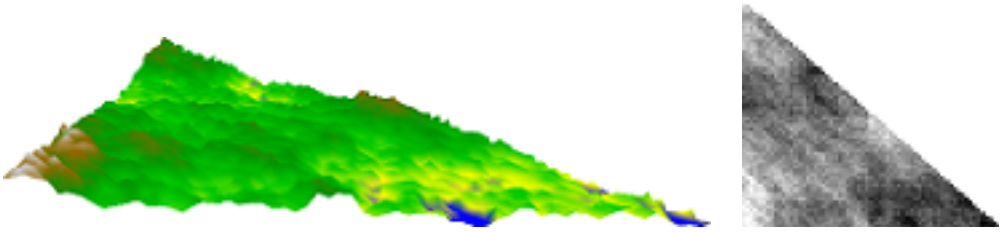
FIG. 3 – Algorithme Mid-point displacement à base triangulaire : Initialement les sommets du triangle n'ont pas de valeur. Une valeur aléatoire est attribuée aux 3 sommets lors de l'initialisation. Ensuite chaque itération divise chaque triangle en 4 triangles en attribuant à chaque milieu des trois côtés une valeur.

La figure 4 montre un terrain généré avec cette méthode.

Fonctionnement

Principe fractal

Dans chaque forme, le facteur de déplacement aléatoire permet de définir



Terrain généré Carte du terrain

FIG. 4 – Terrain généré par l'algorithme Mid-point displacement à base triangulaire

l'intervalle de valeurs du bruit. C'est ce facteur qui à chaque itération permet de bruiter les valeurs, relativement à l'échelle de la forme traitée. Ce facteur est fixé initialement à la valeur de rugosité ("smoothness") puis multiplié par $2^{-smoothness}$ à chaque itération. Ainsi, le principe fractal apparaît clairement : le même algorithme est appliqué à chaque échelle en s'y adaptant.

Représentation

En base carrée, la représentation est simple : une matrice de côté $2^n + 1$.

En base triangulaire, j'ai choisi de représenter le terrain à générer par un tableau de 2^n+1 lignes où la i^e ligne ($i \in [0, 2^n]$) a pour longueur $i+1$ (Figure 5).

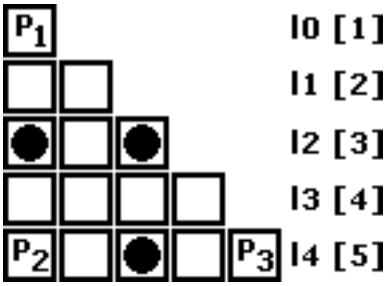


FIG. 5 – Représentation mémoire ($dtail = 2, hauteur = 2^2 + 1 = 5$). Les points noirs sont les emplacements des sommets calculés à la première itération.

Implémentation

En ce qui concerne maintenant l'implémentation, ces algorithmes sont la

plupart du temps présentés de manière récursive. Il est vrai que leur définition présente un aspect récursif immédiat. Cependant, ces algorithmes sont traitables de manière itérative. En effet, au lieu de répéter le processus de génération à chaque sous partie donnée en paramètre, il est possible de parcourir l'ensemble selon un pas se réduisant à chaque étape. Cela revient à effectuer séquentiellement chaque étape de récursivité de même profondeur. Ainsi, bien que moins lisible et intuitive, cette méthode nous permet de générer avec des ressources et un temps plus raisonnables un même résultat.

Cette méthode offre également un second avantage très intéressant : elle résout de manière transparente le problème de la sensibilité contextuelle. Dans une approche récursive, chaque sous partie ne possède aucune information sur ses voisines, cependant les sommets qu'elles ont en commun doivent être de même hauteur, de même que certains calculs (phase "square" de la forme à base carrée) nécessitent de connaître la valeur de certains points des sous parties voisines. Pour la forme récursive de la base triangulaire, il faut soit fournir de l'information sur les voisins, soit contourner cela en biaisant le générateur aléatoire pour qu'il génère toujours la même valeur pour les mêmes points parents. Pour la base carrée, il faudra toujours préciser au moins certaines informations sur les voisins dont on dépend pour calculer des points. Or, quelle que soit la base dans notre approche itérative, on traite chaque point l'un après l'autre, une et une seule fois. De plus, la totalité de l'information est toujours accessible. Le problème de sensibilité contextuelle y est donc réglé par construction.

1.1.2 Algorithme "Spectral synthesis"

Le principe de cette méthode est de voir le terrain comme la superposition de plusieurs couches de détail de plus en plus fin. Pour chaque couche, une fréquence distribue des valeurs sur certains points. Ensuite, une interpolation est faîte pour lisser les points situés entre ces sommets. Nous ne discuterons pas ici des différents types d'interpolation et prendrons à ce titre l'interpolation linéaire. La première couche possède une fréquence basse et génère la forme globale du terrain. Cette forme est ensuite affinée à chaque itération par la superposition de couches de fréquences de plus en plus élevées.

A chaque itération, pour une fréquence f, un tableau de $f \times f$ valeurs est généré. Ces valeurs ont pour amplitude maximale

$$A = p^i$$

où p est la "persistance" et i la i^e itération en cours. A chaque passe (ou itération), la fréquence est multipliée par un pas :

$$f_{i+1} = f_i \times step$$

Ainsi, chaque itération apporte de la précision sur la forme déjà en place du terrain, en respectant le principe fractal par le biais de l'amplitude et de l'augmentation de fréquence. La figure 6 montre l'évolution d'un terrain au fil des itérations.

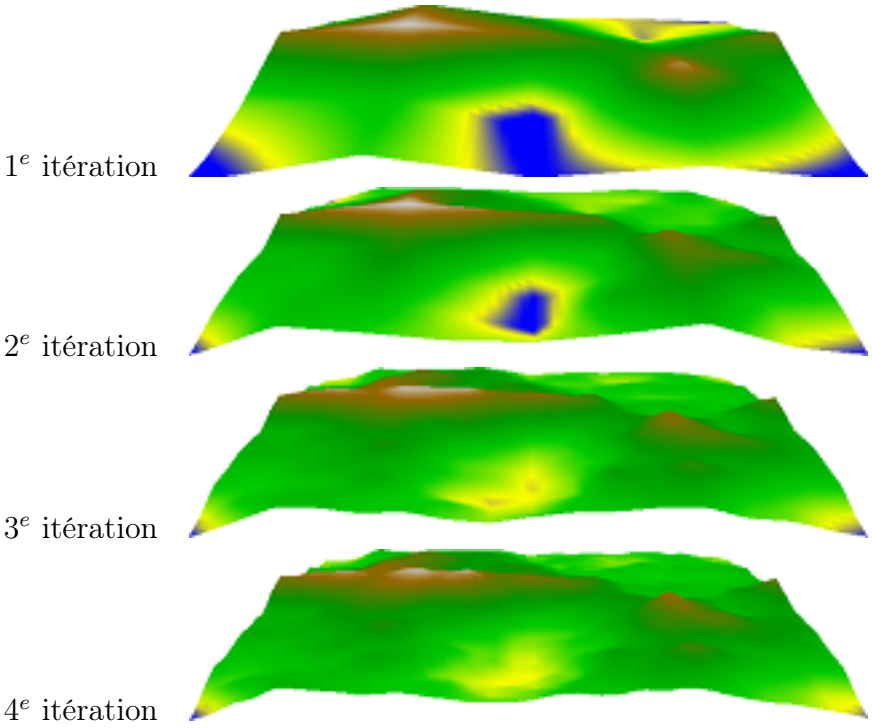


FIG. 6 – Algorithme Spectral synthesis : Evolution d'un terrain de la première à la quatrième itération avec une fréquence initiale de 4 (points par ligne et par colonne), un multiplicateur fréquentiel de 2 par itération, et une persistance de 0,5. Le type d'interpolation utilisée ici est une interpolation linéaire, elle est simple, rapide et donne de bons résultats.

1.2 Algorithme à base de diagrammes de Voronoï

Jusqu'ici, les méthodes que nous avons abordées donnent des résultats réalistes au niveau de l'aspect du sol et des variations d'altitudes, mais le terrain reste très homogène. Le principe de génération n'offre pas de terrains très vallonnés. Le relief est assez constant.

La synthèse à l'aide de diagrammes de Voronoï va nous permettre de produire des terrains au contraire très valonnés mais à pente linéaire. Le

réalisme fractal de la rugosité est donc mis de côté ici pour introduire un nouveau type de terrain.

Le principe est de choisir aléatoirement des points dans l'espace à générer. Ce seront les points de repère pour la génération du diagramme de Voronoï. Afin d'attribuer une hauteur à chaque point de l'espace, on doit introduire la notion de distance d'un point à un autre. Soit d la fonction qui calcule la distance entre deux points p_1 et p_2. Cette distance est défini par

$$d(p_1, p_2) = (p_2.l - p_1.l)^2 + (p_2.c - p_1.c)^2$$

où $p.l$ est la ligne de p et $p.c$ sa colonne.

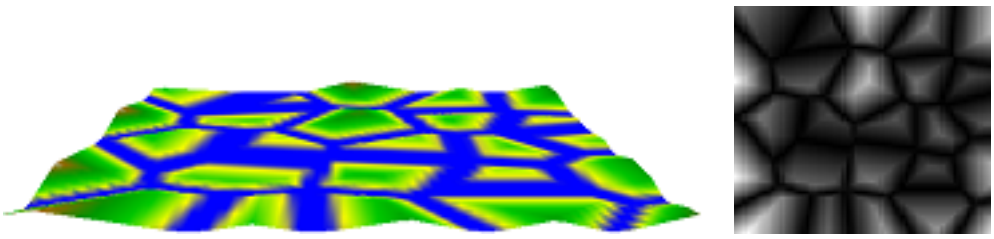
La distance euclidienne, qui prend la racine carré de cette distance, n'est pas utilisée ici car ce calcul supplémentaire apporte peu au rendu et prends un temps conséquent.

De là, la hauteur de p se définit par :

$$h(p_{l,c}) = c_1 \times d(p_{l,c}, 1^{er}Ppp) + c_2 \times d(p_{l,c}, 2^{e}Ppp) + \ldots + c_n \times d(p_{l,c}, n^{e}Ppp)$$

où c_n est un poids attribué à chaque distance et i^ePpp le i^e plus proche point de repère par rapport à p.

Pour générer des terrains montagneux, connaître c_1 et c_2 est suffisant et donne de bons résultats pour $c_1 = -1$ et $c_2 = 1$. Ainsi, on allège le calcul en ne recherchant que les deux plus proches points de repère du point considéré. La figure 7 montre un terrain généré dans ces conditions. Cependant, bien que vallonné, ce terrain est à pente très linéaire. Nous verrons plus loin comment conserver les propriétés des diagrammes de Voronoï en combinant cette technique avec d'autres pour obtenir des terrains de même topologie avec une bonne rugosité.



Terrain généré Carte du terrain

FIG. 7 – Diagramme de Voronoï : terrain généré avec $c_1 = -1$ et $c_2 = 1$

2 Amélioration de terrain par combinaison

Bien que simple et basique sur le résultat, la phase de génération du terrain de base est très importante. En effet, plus le terrain de base est proche du terrain désiré, plus le travail de retouche est réduit par la suite.

Les algorithmes cités plus haut donnent des résultats encourageants mais il faudrait combiner certaines techniques pour réussir à garder le plus d'avantages et le moins d'inconvénients de chacune. Nous allons donc voir comment améliorer les résultats en combinant certaines techniques avec d'autres.

Nous allons aborder la technique présentée dans [1] permettant de créer des lits de rivières, puis une seconde citée dans [2] permettant de générer des terrains vallonnés mais également irréguliers.

2.1 Algorithme "Squig-curves"

L'article [1] nous présente une manière fractale de générer un lit de rivière dans un terrain à base triangulaire. Son objectif est de proposer une méthode permettant de coupler, en un seul algorithme, la génération élémentaire par "Mid-point displacement" à base triangulaire, à l'algorithme "Squig-curves" permettant la génération fractale de rivières. Il explique le principe de modification du premier algorithme pour prendre en compte les contraintes imposées par le second.

Dans un premier temps, je vais exposer le principe de "Squig-curves", puis j'expliquerai l'idée de Przemyslaw Prusinkiewicz et Mark Hammel.

2.1.1 Principe de base

L'idée de base est de définir dans un triangle un côté d'entrée et un côté de sortie de la rivière. Ensuite, diviser ce triangle en quatre sous parties égales, les trois nouveaux sommets étant les milieux des trois segments. Choisir aléatoirement pour le côté d'entrée un de ses deux segments qui deviendra la nouvelle entrée. Faire de même avec la sortie. Mettre à jour parmi les nouveaux segments internes les côtés entrée et sortie. La figure 8 montre plusieurs divisions possibles et la figure 9 montre un lit de rivière ainsi généré.

2.1.2 Combinaison avec "Mid-point displacement"

Compte tenu de la structure triangulaire de l'algorithme "Squig-curves", il parait assez intuitif de le coupler avec un algorithme de génération à même base. L'objectif de [1] est donc de nous monter comment intégrer cet algorithme au "Mid-point displacement" à base triangulaire.

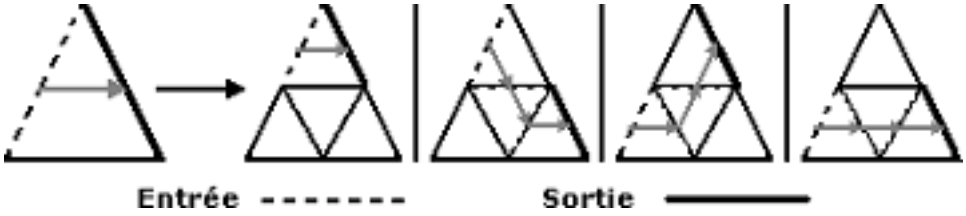
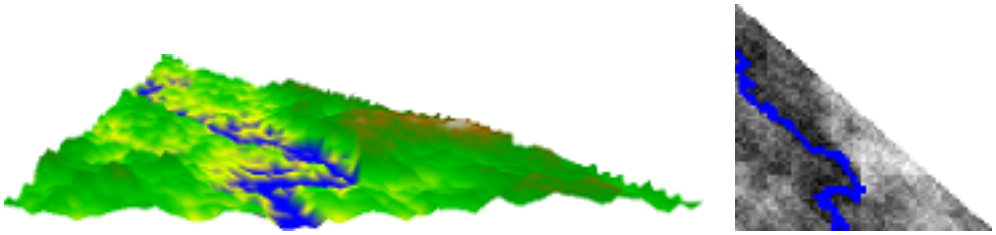


FIG. 8 – Différentes divisions de parcours possibles à partir d'une situation.



Terrain généré Carte du terrain

FIG. 9 – Lit de rivière généré sur un terrain de détail 6

Le principe est d'effectuer un "Mid-point displacement" normal mais de changer les instructions dans le cas particulier où un côté de triangle est un passage de rivière. Dans ce cas, il n'est plus possible d'attribuer des valeurs aléatoires aux sommets. Pour respecter l'écoulement de l'eau, il faut au contraire garantir que la hauteur du milieu du segment de sortie est au plus égale à la hauteur du milieu de celui d'entrée. De plus, il faut garantir que les sommets voisins n'ont pas une hauteur inférieure. Pour ce faire, il faut donc prendre au plus la valeur minimale des sommets voisins ne faisant pas passer la rivière. C'est cette valeur qui sera choisie dans cet algorithme.

2.1.3 Limites

Constance de l'altitude L'algorithme proposé, se basant uniquement sur une altitude maximale à ne pas dépasser, ne permet que de produire des rivières à pente très douce. Il produit des rivières de vallées, aucune cascade ou cours d'eau rapide ne peut être généré par cette algorithme.

Vallées asymétriques Le fait de prendre comme valeur, au passage d'une rivière, le minimum des altitudes voisines peut facilement créer une falaise d'un côté et une pente douce en face. En effet, si l'amplitude d'altitude des voisins d'un point de passage est grande, la valeur minimum qui lui sera at-

tribuée peut être très différente de la valeur d'un de ses voisins à altitude élevée. Cela peut donc conduire à une rupture brutale de la continuité d'altitude d'un côté et pas de l'autre. Or dans la nature, on trouve plus facilement une symétrie des côtés.

Pas d'affluents ni de confluents L'algorithme proposé ne permet pas de joindre deux lits de rivière ou d'en diviser un. De ce fait, la rivière produite ne possède ni affluents ni confluents. Cependant, dans l'idée, il semble possible de posséder plusieurs rivières en construction, et donc d'appliquer la méthode à plusieurs endroit de la grille. Il faudrait alors prévoir un cas de jonction ou de séparation. L'article [1] montre le résultat attendu par ce type d'amélioration.

2.2 Combinaison Diagrammes de Voronoï et Diamond-square

La méthode de génération par diagrammes de Voronoï nous permet de créer un terrain très vallonné, là où les fonctions à bruit y parviennent mal, et ces fonctions à bruit offre une surface irrégulière, là où les diagrammes de Voronoï ne fournissent que du linéaire. En combinant ces deux techniques, il est possible de générer un terrain possédant les caractéristiques voulues de chaque méthode sans trop hériter de leurs inconvénients.

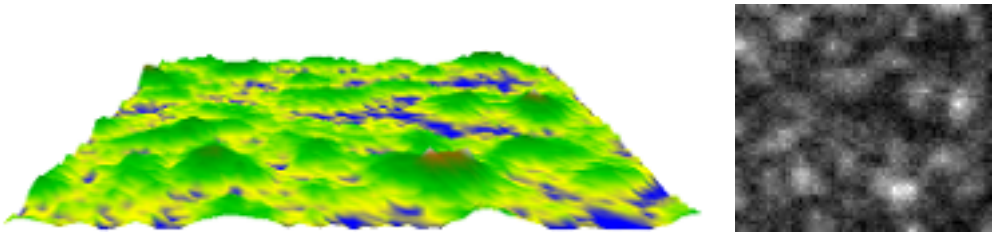
Le principe ici est d'utiliser la technique des diagrammes de Voronoï pour générer certains points, puis d'appliquer celle du mid-point displacement pour générer les autres.

On donne pour cela un pas *step*, et tous les *step* points (horizontalement et verticalement), on applique la méthode des diagrammes de Voronoï. Ensuite, les carrés définis par ces points valués sont remplis par mid-point displacement. Il y a cependant une contrainte sur la taille du terrain. Pour appliquer mid-point displacement, il faut un côté de taille $2^{detail} + 1$, et pour appliquer le pas et diviser ainsi la matrice en sous matrices de taille type $2^n + 1$, il faut vérifier 2^{detail} $modulo$ $step = 0$.

La figure 10 montre un terrain généré par cette méthode. On reconnaît la structure des terrains générés avec la méthode des diagrammes de Voronoï mais ce terrain est maintenant beaucoup plus irrégulier et ainsi plus réaliste.

3 Amélioration de terrain par retouche

Dans la partie précédente, nous avons introduit deux techniques permettant de générer un premier terrain plus proche du réel que ceux générés avec les méthodes de base. Cependant, cette génération est encore trop limitée



Terrain généré Carte du terrain

FIG. 10 – Diagramme de Voronoï avec Mid-point displacement : terrain généré avec $c_1 = -1$, $c_2 = 1$ et une rugosité de 0.5

pour être vraiment acceptable, par exemple à l'échelle d'un jeu de stratégie. Afin d'obtenir un terrain relativement réaliste, varié et praticable, il faut retoucher ce terrain généré. Pour cette phase, comme il est dit dans [2], il est très important d'avoir travaillé au préalable la technique de génération de manière à avoir le moins de retouches à faire. Toutes ces méthodes sont en effet très lourdes dès qu'il s'agit de générer de grands terrains. La qualité d'un terrain généré se jugera donc en fonction du temps de retouche nécessaire.

3.1 Retouche par Synthèse spectrale

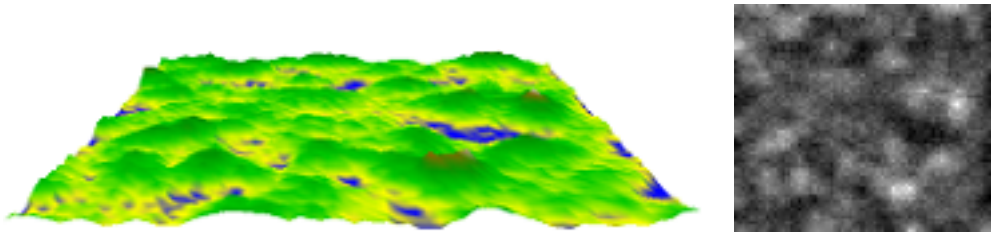
Cette méthode simple de retouche consiste à appliquer une ou plusieurs couches de l'algorithme Spectral synthesis sur un terrain généré au préalable. Les résultats obtenus par la combinaison diagrammes de Voronoî et mid-point displacement ayant un très bon rapport variation/réalisme en font la base idéale pour appliquer une synthèse spectrale.

Cette technique ne présente rien de nouveau mais donne un résultat nettement meilleur que la simple combinaison diagrammes de Voronoï et mid-point displacement.

La figure 11 montre l'amélioration apportée par la synthèse spectrale sur le terrain de la figure 10.

3.2 Retouche par Perturbation

Cette deuxième méthode à pour objectif d'introduire du désordre dans un terrain généré. Elle consiste à déplacer dans un voisinage réduit les points de manière à rompre la continuité du relief. De même que précédemment, nous l'appliquerons sur un terrain généré avec diagramme de Voronoï et mid-point displacement. Le désordre est régulé à l'aide d'un unique paramètre appelé "magnitude" qui donne le rayon de déplacement autorisé d'un point. Cette

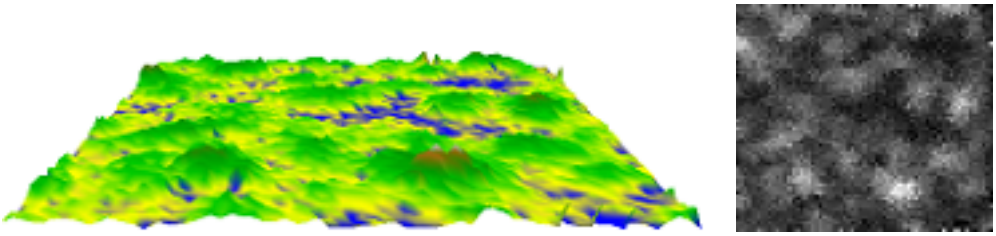


Terrain généré Carte du terrain

FIG. 11 – Diagramme de Voronoï avec Mid-point displacement puis Synthèse spectrale

technique ne vas en général pas apporter un résultat très esthétique ni très réaliste mais va cependant pouvoir apporter une rupture brutale du relief. Le terrain ainsi obtenu sera prêt pour la dernière étape : la retouche par érosion.

La figure 12 montre le changement apporté par la perturbation sur le terrain de la figure 10.



Terrain généré Carte du terrain

FIG. 12 – Diagramme de Voronoï avec Mid-point displacement puis Perturbation

3.3 Simulation d'érosion

Dans [2], Jacob Olsen présente une approche originale, inspirée de la physique, la simulation d'érosion. L'objectif est toujours de générer des terrains au plus proche du réel mais selon une méthode et des objectifs différents. Dans son article, il oriente l'objectif de la génération de terrain vers le jeu, plutôt de stratégie. Il va donc rechercher certains critères de topologie, indispensable à ce type d'application. Les méthodes par érosion vont agir sur certaines configurations du terrain, afin de corriger certaines structures indésirables, pour rendre le terrain réaliste et utilisable. Elle lui applique une "érosion" qui va adoucir certaines pentes tout en conservant sa topologie.

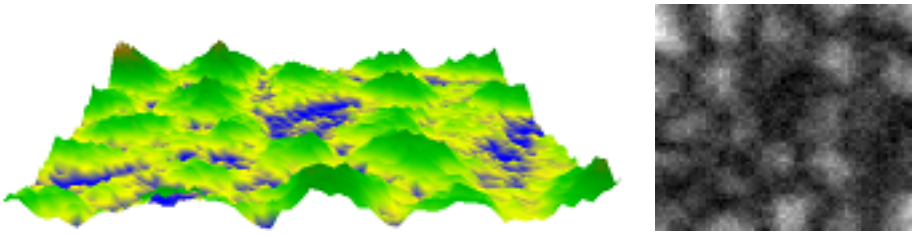
Ainsi, si le terrain de base à une topologie intéressante, la méthode par érosion prend en charge de le rendre réaliste et praticable pour le jeu.

Le principe est relativement simple, il s'agit de comparer la différence de niveau entre deux points voisins et de déplacer de la matière du point culminant vers l'autre. Pour un point de référence p et un voisin $p_{i,j}$, la différence de niveau s'exprime donc par :

$$d(p, p_{i,j}) = h(p) - h(p_{i,j})$$

Deux types d'érosions sont proposés dans l'article : l'érosion thermique ("Thermal erosion") et l'érosion hydrolique ("Hydraulic erosion"). Le premier type représente un vieillissement géologique du terrain où le déplacement de matière peut être assimilé à des éboulements dus à l'usure. Le second type simule l'action de l'eau, détrempant le sol, et entrainant de la matière en suspension avant de la fixer ailleurs.

Le terrain de référence pour les comparaisons d'algorithmes d'érosion est donné Figure 13.



Terrain de base | Carte des niveaux

FIG. 13 – Terrain de base utilisé comme référence pour les algorithmes à érosion

3.3.1 Erosion Thermique

Principe

Ce type d'érosion a pour but d'aplanir le terrain, aussi va-t-il s'appliquer aux couples de points ayant une différence de niveau supérieure à un certain seuil noté T. Ce seuil se définit par :

$$T = \frac{k}{N}$$

où N est la longueur du côté du terrain dans le cas d'un terrain carré. k est un paramètre qui peut être fixé ou fourni par l'utilisateur. Les valeurs

intéressantes de k se situent globalement entre 2 et 20. Il est important de noter que ce type d'érosion va adoucir les pentes raides et que les pentes douces (inférieures à T) ne subirons pas d'érosion. Cet algorithme va donc aplatir le terrain.

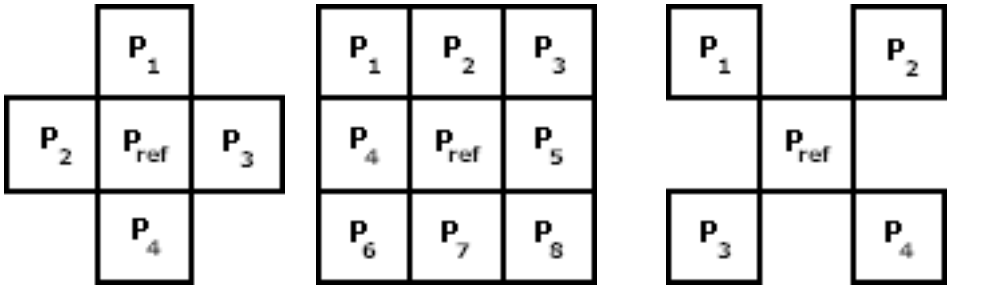
Un deuxième paramètre important, noté c, va déterminer la quantité de matière déplaçable sur un couple à chaque itération. Ainsi, le déplacement de matière qui a lieu lorsque $d(p, p_{i,j}) > T$ se traduit par $c(d(p, p_{i,j}) - T)$.

Le transfert de matière d'un point vers un autre se traduit sur la hauteur comme suit :

$$h(p_{i,j}) = \begin{cases} h(p_{i,j}) + c(d(p, p_{i,j}) - T) & \text{si} \quad d(p, p_{i,j}) \geq T \\ h(p_{i,j}) & \text{sinon} \end{cases}$$

Cependant, le voisinage d'un point ne se limite pas à un seul autre point. Plusieurs types de voisinages peuvent être utilisés (Figure 14) :

- Von Neumann,
- Moore,
- Von Neumann avec rotation.



Von Neumann Moore Von Neumann Rotationné

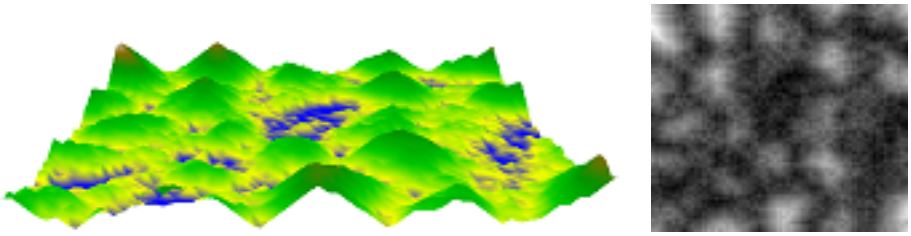
FIG. 14 – Les trois types de voisinage proposés du point p_{ref}

Aussi, le déplacement de matière n'est plus tout à fait aussi trivial. Sur la quantité de matière déplaçable $c(d(p, p_{i,j}) - T)$, chaque point le nécessitant va récupérer une fraction de cette quantité. Cette fraction se définit par : $\frac{d(p,p_{i,j})}{d_{total}}$ où d_{total} est la somme des distances supérieures à T. Pour un voisin $p_{i,j}$ de p, la nouvelle hauteur est alors :

$$h(p_{i,j}) = h(p_{i,j}) + c(d_{max} - T) \times \frac{d(p, p_{i,j})}{d_{total}}$$

où d_{max} est la plus grande distance $d(p, p_{i,j})$ supérieure à T.

La Figure 15 montre l'action de l'algorithme de référence de l'érosion thermique.



Terrain érodé Carte des niveaux

FIG. 15 – Effet du filtre d'érosion thermique avec 100 itération, $c = 0.5$ et $k = 4$

Optimisations

Ce type d'algorithme, moyennant quelques hypothèses, peut être grandement accéléré. Jacob Olsen présente quelques points sur lesquels il est possible de faire de telles hypothèses :

- Choix du voisinage : En prenant un voisinage de Von Neumann au lieu d'un voisinage de Moore, on explore deux fois moins de voisins.
- Distribution : Plutôt que de distribuer un peu de matière à plusieurs points, il propose de tout donner à celui qui à la différence de hauteur la plus élevée (le point le plus en contrebas).
- Maximisation du transfert de matière : Ce tranfert par itération peut être maximisé à $\frac{d_{max}}{2}$.
- Tolérer la modification directe sur la carte du terrain, sachant que toute modification d'un point peut avoir des conséquences sur le traitement d'un prochain point. Cette réaction en chaîne peut, si l'on distribue à plusieurs voisins, créer un effet d'ondelette. Dans le cas ici où l'on ne distribue qu'au plus nécessiteux, cet effet de bord n'a pas lieu.

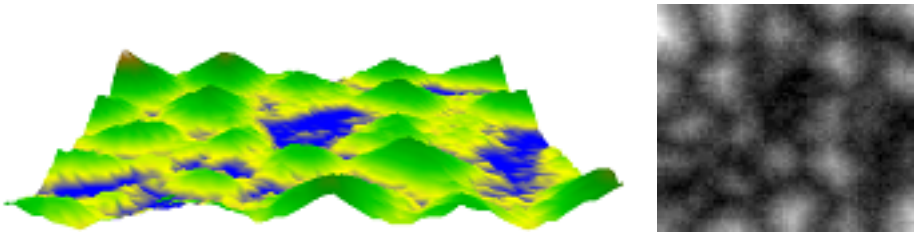
Ainsi, Jacob Olsen, suite à ses travaux, propose une implémentation rapide de ce type d'érosion, très proche du modèle original. Pour des explications précises et des expérimentations comparées, consulter son article [2].

La Figure 16 montre l'action de l'algorithme optimisé de l'érosion thermique.

3.3.2 Erosion Hydraulique

Principe

Jacob Olsen propose également un autre type d'érosion, basé sur l'action de l'eau, issue par exemple de la pluie. De l'eau pleut sur le terrain, une couche de ce terrain s'y dissout, puis l'eau ruisselle entraînant avec elle un peu de matière dissolue. A l'inverse de l'algorithme précédent, celui-ci va



Terrain érodé Carte des niveaux

FIG. 16 – Effet du filtre optimisé d'érosion thermique avec 100 itération et $k = 4$

s'appliquer aux petites pentes et va combler les petites cavités.

Cet algorithme se décompose en quatre phases :

- Arrivée de l'eau,
- Dissolution de matière dans l'eau (sédimentation),
- Transfert de l'eau et des sédiments,
- Evaporation de l'eau et dépôt de sédiments.

Tout d'abord, il doit pleuvoir sur le terrain. Afin de connaître l'état du terrain en terme d'eau, une carte w des niveaux de l'eau est dressée en parallèle et remplie comme suit :

$$w_{i,j} = w_{i,j} + K_r$$

où K_r est la quantité d'eau ajoutée à une cellule par itération.

Puis, cette eau dissout de la matière, laquelle sera stoquée dans une carte des sédiments m. Cette dissolution s'exprime par une diminution du niveau du point et une augmentation de sa contenance en sédiments :

$$h_{i,j} = h_{i,j} - K_s \times w_{i,j}$$

$$m_{i,j} = m_{i,j} + K_s \times w_{i,j}$$

où K_s est la constante de solubilité du terrain. En d'autre terme, elle définit la quantité de matière qui va se dissoudre dans l'eau.

Ensuite, le transfert peut opérer. On définit l'altitude a d'un point par $a = h + w$. Le but est de niveler les voisins $p_{i,j}$ de p au même niveau, soit : $a(p) = a(p_{i,j})$. Le transfert d'eau $\Delta w_{i,j}$ de p vers $p_{i,j}$ est de :

$$\Delta w_{i,j} = min(w, \Delta a) \times \frac{d_{i,j}}{d_{total}}$$

avec $\Delta a = a - \bar{(}a)$ et $\bar{a}$ l'altitude moyenne des points (référence et voisins) impliqués dans l'opération. Le transfert de sédiments $\Delta m_{i,j}$ est alors :

$$\Delta m_{i,j} = m \times \frac{\Delta w_{i,j}}{w}$$

Enfin, le transfert effectué, l'eau s'évapore un peu en chaque point suivant la loi :

$$w = w \times (1 - K_e)$$

où K_e est le coefficient d'évaporation. Puis, selon la quantité de sédiments et d'eau présents en chaque point, une partie des sédiments peut se retransformer en terrain. L'eau conserve une quantité de terrain au plus égale à m_{max} définit par :

$$m_{max} = K_c \times w$$

où K_c est la capacité de contenance sédimentaire de l'eau.
La quantité de sédiments se retransformant en terrain est donc :

$$\Delta m = max(0, m - m_{max})$$

On met alors à jour m et h :

$$m = m - \Delta m$$

$$h = h + \Delta m$$

Les constantes de terrains utilisées pour les exemples de génération sont :

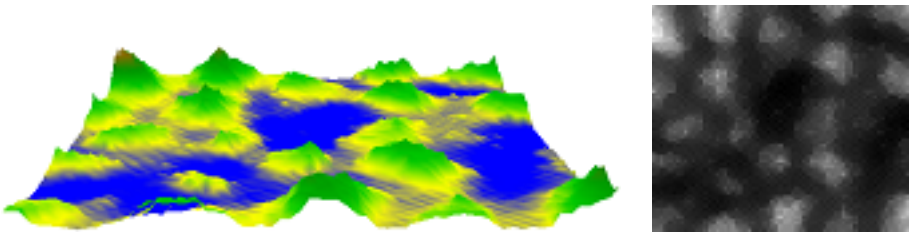
$$K_r = 0.01$$

$$K_s = 0.01$$

$$K_e = 0.5$$

$$K_c = 0.01$$

La Figure 17 montre l'action de l'algorithme de l'érosion hydraulique.



Terrain érodé — Carte des niveaux

FIG. 17 – Effet du filtre d'érosion hydraulique

Optimisations

Ce type d'algorithme, moyennant quelques hypothèses, peut être grandement accéléré. Jacob Olsen présente quelques points sur lesquels il est possible de faire de telles hypothèses.

- Choix du voisinage : En prenant un voisinage de Von Neumann au lieu d'un voisinage de Moore.
- Distribution : L'eau n'est transférée qu'au voisin le plus bas.
- Fusion Eau/Sédiments : En posant $K_s = K_c$, la quantité de sédiments transportée par l'eau est égale à la quantité de sédiments créée par l'eau. Ainsi, la carte des sédiments devient obsolète.
- Tolérer la modification directe sur la carte du terrain et de l'eau (ou des sédiments, les deux ayant fusionnées).

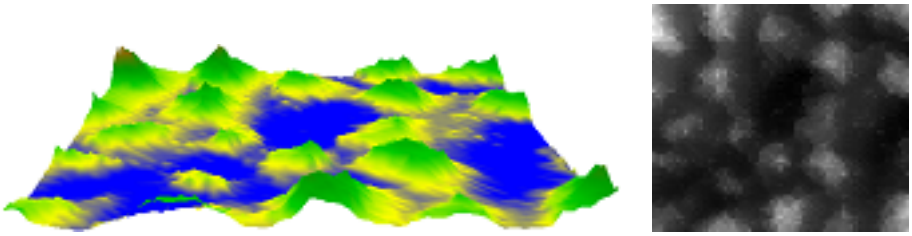
Ces optimisations nous amènent à réécrire certaines quantités :

$$a(p_{i,j}) = h(p_{i,j}) + m(p_{i,j})$$

Cette modification de l'algorithme permet un gain de temps de facteur cinq environs pour un résultat presque identique.

Cependant, ces algorithmes nécessitent beaucoup d'itérations, de l'ordre de la centaine, pour effectuer ce que l'on attend.

La Figure 18 montre l'action de l'algorithme de l'érosion hydraulique.



Terrain érodé — Carte des niveaux

FIG. 18 – Effet du filtre optimisé d'érosion hydraulique

3.3.3 Nouvel algorithme d'Erosion

Combinaison des érosions thermique et hydraulique

Dans son article, Jacob Olsen cherche à générer des terrains utilisables pour des jeux de stratégie. Aussi, il définit des critères caractérisant un bon terrain de jeu. Ce terrain doit permettre à des unités de se déplacer à travers, donc il faut un ensemble de cellules voisines de pente douce et de largeur suffisante, représentant une bonne part du terrain. Il faut de plus dans cet ensemble

de cellules accessibles des endroits plats pour poser des bâtiments. ces seuls critères tendent à définir un espace plat comme idéal, il faut donc également un critère assurant un certain relief dans le terrain. Ce critère est appelé score d'érosion ϵ. Il se définit comme tel :

$$\epsilon = \frac{\sigma_s}{\bar{s}}$$

où $\bar{s}$ est la moyenne des pentes du terrain, donc des différences de hauteur entre tous les points deux à eux voisins, et σ_s est la moyenne des écarts des pentes à la moyenne.

Un bon terrain, laissant se déplacer au mieux des unités doit donc posséder une moyenne $\bar{s}$ basse, et afin de garantir du relief doit avoir un σ_s élevé. L'objectif est donc d'obtenir un score d'érosion élevé.

D'une part, l'érosion thermique, étalant le terrain, diminue $\bar{s}$ mais diminue aussi σ_s. Cet algorithme, notamment pour sa forme optimisée, est par contre très rapide. D'autre part, l'érosion hydraulique, comblant les petites irrégularités, diminue $\bar{s}$ en augmentant σ_s. Cependant, cet algorithme, même optimisé reste très lent. L'érosion hydraulique demande beaucoup trop d'itérations.

Le principe recherché va donc être de combiner au possible la vitesse de l'érosion thermique au haut score d'érosion de l'érosion hydraulique. L'idée de Jacob Olsen va être de reprendre l'algorithme optimisé de l'érosion thermique mais de l'appliquer aux petites pentes, plutôt qu'aux grandes. Autrement dit, les déplacement de matière auront lieu non pas pour

$$d_{i,j} > T$$

mais pour

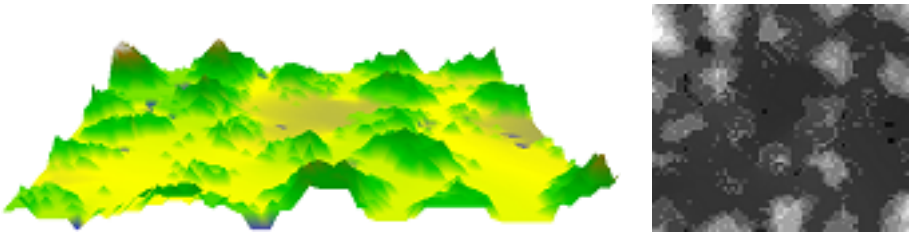
$$d_{i,j} <= T$$

Ainsi, la structure efficace de l'algorithme optimisé d'érosion thermique est conservée, mais le transfert est appliqué, comme pour l'érosion hydraulique, sur les petites pentes. Ainsi, le terrain obtenu tend à conserver ses reliefs sans les petites cavités qui peuvent gêner une construction ou un déplacement.

La Figure 19 montre l'action de ce nouvel algorithme d'érosion.

Modification du terrain de base

Pour obtenir un terrain de base mieux disposé à posséder un bon score d'érosion, il est aussi possible de biaiser la génération en îlots du diagramme de Voronoï en construisant en parallèle une carte des régions, où une région est constituée de tous les points ayant le même plus proche point de repère.



Terrain érodé Carte des niveaux

FIG. 19 – Effet du filtre nouvelle optimisation d'érosion thermique

Ensuite, une même altitude est attribuée à tous les points d'une même zone. On obtient plusieurs zones séparées par une rupture d'altitude. Ensuite, en effectuant le produit point à point des deux cartes, des îlots vont s'affaisser, d'autres non, et les écarts entre les îlots vont s'agrandir. En choisissant des valeurs binaires pour la carte des régions, le produit va supprimer certains ilôts et en garder d'autres. Ainsi, le terrain produit peut posséder de larges passages en contrebas. Dans le cadre des terrains de jeu, dont l'article [2] traite, ce type de biais pour la génération de base s'avère contribuer à l'apport de propriétés fondamentale pour le terrain jeu.

Conclusion

A travers ces quelques procédés de génération, il apparait qu'il semble difficile de définir une façon empirique de générer un terrain. En effet, selon le type de terrain voulu, différentes méthodes, optimisations ou combinaisons seront requises. De plus, ces méthodes relativement simples, sont déjà très lourdes en calcul et il ne semble pas possible pour du temps réel de rajouter à ces algorithmes des calculs supplémentaires. Il me semble donc incontournable de poser des critères d'exigence qui permettront de faire un choix de techniques. De même, pour générer un terrain très grand possédant plusieurs types de terrains (par exemple montagne, rivière, lacs, champs plats...), il faudra appliquer différentes techniques à différents endroits de la carte. Ainsi, bien que des méthodes simples permettent de générer des terrains aux propriétés réalistes, la limite de la diversité offerte se fait très vite sentir.

L'approche inspirée du comportement naturel me semble cependant un bon choix dans ce type de travail. L'approche fractale, modélisant bien certains aspects de la nature, a apporté pour la génération de terrains et de végétation des possibilités nouvelles et des résultats très réalistes. La simulation d'érosion, exemple de l'inspiration géophysique, apporte également de

bonnes propriétés aux terrains. Je pense qu'une poursuite dans cette voie pouvant peut-être partir de la simulation de l'action tectonique pour appliquer ensuite successivement des acteurs géologiques naturels, comme le vent, la mer, etc, peut amener à synthétiser une éventualité du réel. Ceci aussi bien dans le cadre de la simulation voire de la prévision, que dans le cadre du jeu.

Références

[1] Przemyslaw Prusinkiewicz et Mark Hammel, "A Fractal Model of Mountains with Rivers", Departement of Computer Science, University of Calgary, Calgary, Alberta, Canada T2N 1N4, extrait de *Proceeding of Graphics Interface '93*, pages 174-180, May 1993.

[2] Jacob Olsen, "Realtime Procedural Terrain Generation, Realtime Synthesis of Eroded Fractal Terrain for use in Computer Games", Departement of Mathematics And Computer Science (IMADA), University of Southern Denmark, October 31, 2004.

[3] http ://www.vterrain.org

Implémentation

Les exemples de terrains générés par les méthodes présentées ici sont issus du programme "Terrain Rendering" (Figure 20) que j'ai développé à l'occasion de ce travail. Il est écrit en Java et utilise la librairie Java3D de Sun pour le rendu. Il est téléchargeable librement sur mon site http ://rone56.free.fr.

Il implémente les méthodes de base :

- Mid-point displacement
  - à base carrée,
  - à base triangulaire,
- Spectral synthesis,
- Diagrammes de Voronoï,

les combinaisons :

- Diagrammes de Voronoï avec Mid-point displacement,
- Squig-curves avec Mid-point displacement,

et les retouches :

- par Perturbation,
- par Spectral synthesis,
- par Erosion
  - thermique de référence,
  - thermique optimisée,
  - hydraulique de référence,
  - hydraulique optimisée
  - thermique et hydraulique remaniées.

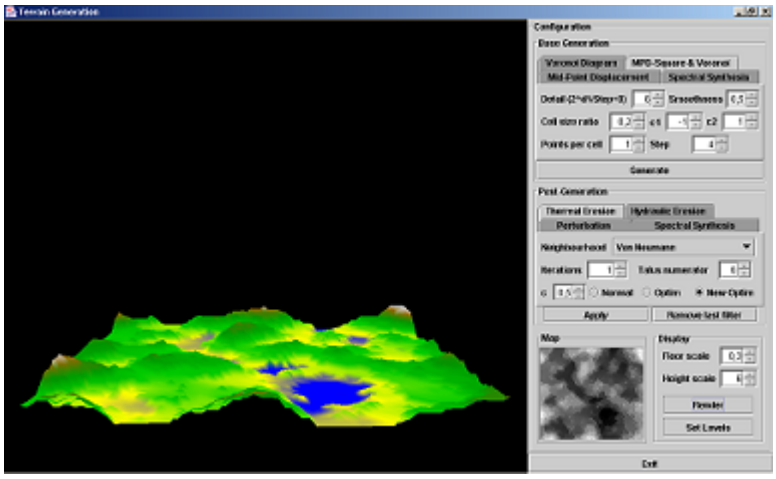


FIG. 20 – Application Java Terrain Rendering