Modélisation de Mondes Virtuels

Arthur Pradier
Janvier 2023

1 Introduction

Ce projet avait pour objectif la mise en place d'une base de code permettant de créer et d'éditer un terrain en 3D. Avec notre projet, nous avons la possibilité de générer un terrain à l'aide de sommes de bruit, d'appliquer une érosion à ce dernier, puis de placer des points d'intérêts représentant des villes qui pourront être reliées entre elles par la suite.

2 Génération de terrain

Nos terrains ont générés à l'aide de bruit procédural de la bibliothèque FastNoiseLite. Nous utilisons du bruit simplexe, avec du Fractal Brownian Motion pour sommer différents octaves de bruit. le terrain en lui même est défini par un champ de scalaires de taille $nx * ny$, ou chaque élément du champ scalaire définit la hauteur du terrain aux coordonnées (x, y) correspondantes. Ces hauteurs sont des valeurs entre -1 et 1, qu'il est possible de mettre à l'échelle lors de l'affichage. L'échelle du bruit est proportionnelle à la taille de notre grille, pour toujours garder le même terrain et simplement augmenter la précision quand la taille de la grille augmente.

A partir de notre terrain, on peut récupérer toutes sortes de données analytiques utiles, toutes exportables en tant qu'images 2D: carte des hauteurs, de la pente, du laplacien, aire de drainage, humidité.

Pour rendre le terrain un peu plus vivant, nous utilisons une méthode de génération de texture qui peut être appelé à n'importe quel moment par le code chargé de l'affichage. Cette méthode lit chaque case de notre grille de hauteurs et assigne une couleur en fonction de certains paramètres. On utilise principalement la pente pour colorer: à basses hauteurs, une pente élevée signifie qu'il n'y aura pas de végétation, le terrain est marron, là où une pente faible signifie que la végétation peut pousser, on met du vert. Entre nos deux extremums, on fait une interpolation linéaire de couleurs. Pour les fortes hauteurs, on se base aussi sur la pente: la neige n'accrochera pas si la pente est trop élevée. Cette méthode, bien que simpliste, produit des résultats satisfaisants. On pourrait cependant l'améliorer en utilisant une fonction de poids, pour prendre en compte l'humidité du terrain, l'exposition au soleil ou tout autre paramètre jugé intéressant.

L'aire de drainage nous permet de savoir où est ce que l'eau s'écoule, et donc de trouver des réseaux de rivières. Quand l'aire de drainage en un pixel atteint plus d'un certain pourcentage de la valeur maximale, on peint le pixel en bleu. On obtient bien un réseau de rivières relativement grossier, mais satisfaisant.

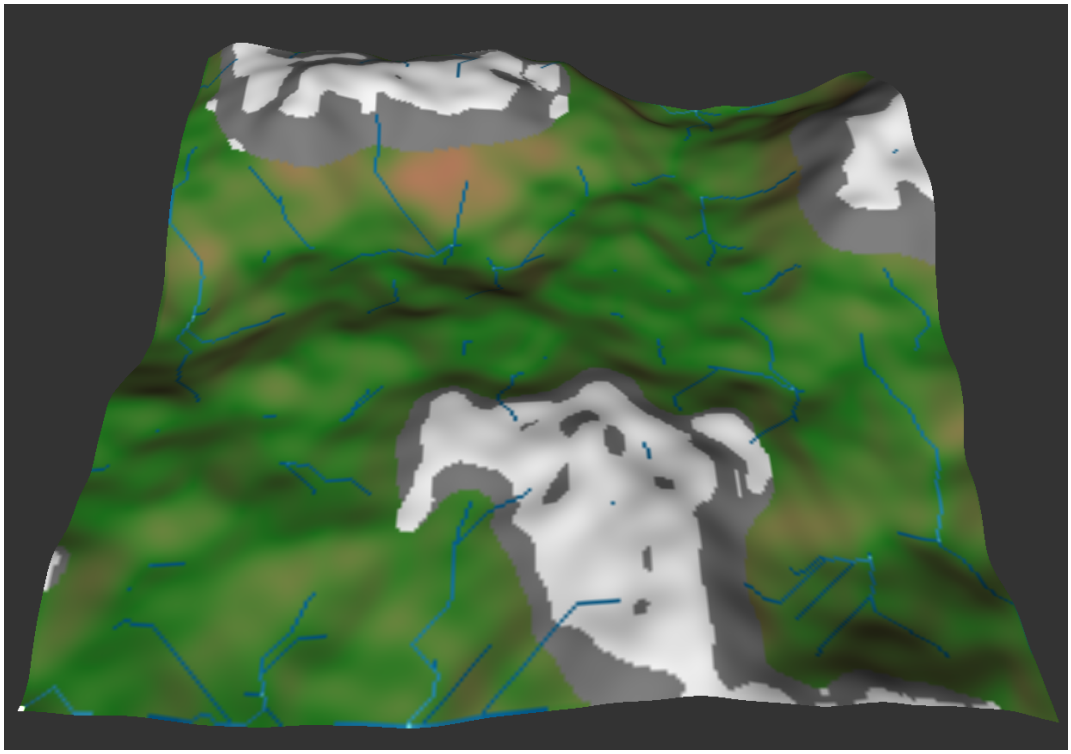

Figure 1: Terrain généré par somme de bruits puis coloré

3 Érosion

Notre algorithme de stream power erosion simplifié [1] permet de modifier notre terrain en suivant un principe relativement simple: les écoulements d'eau creusent le terrain, et la tectonique des plaques le soulève pour créer des montagnes. La formule permettant d'exprimer l'érosion de notre terrain est $u + kA^{n/2}s^n + l\Delta h$, avec u une constante d'uplift, pour exprimer le fait que notre terrain a tendance a être poussé vers le haut. A représente l'aire de drainage, s la pente et Δh le laplacien. k et l sont des coefficients permettant de contrôler nos paramètres.

Dans notre cas, nos coefficients sont très petits, pour éviter un cas ou le point contenant la valeur maximale d'aire de drainage va perdre en hauteur extrêmement rapidement, alors que le reste de la carte reste bouge de manière plus modérée, ce qui résulte en une valeur NaN dans notre champ scalaire (qui rend par conséquent les rendus impossibles). Après 100 itérations de notre algorithme, on obtient de résultat suivant:

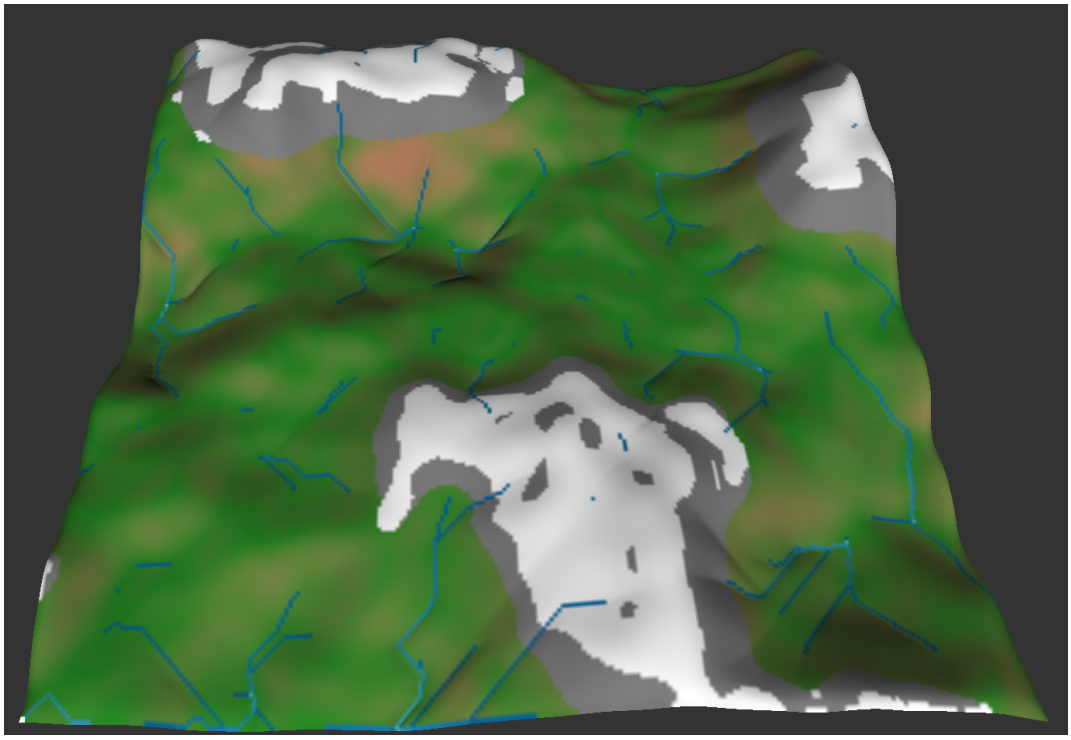

Figure 2: Terrain généré par somme de bruits, coloré puis érodé

4 Placement et croissance de villes

Pour rendre le terrain plus intéressant, nous plaçons et faisons grandir des villes. Lors de la création de notre terrain et lorsqu'il est modifié après érosion par exemple, on appelle une fonction chargée de détecter un nombre n de points d'intérêts, avec n un paramètre passé au constructeur du terrain. La position de ces points d'intérêt dépend d'une fonction de poids. On cherche à trouver un point qui maximise un score prenant en paramètre la hauteur, la pente et l'aire de drainage. Pour faire simple, on veut en endroit le plus haut et le plus plat possible, et le plus proche possible d'un écoulement d'eau sans être littéralement dans l'eau.

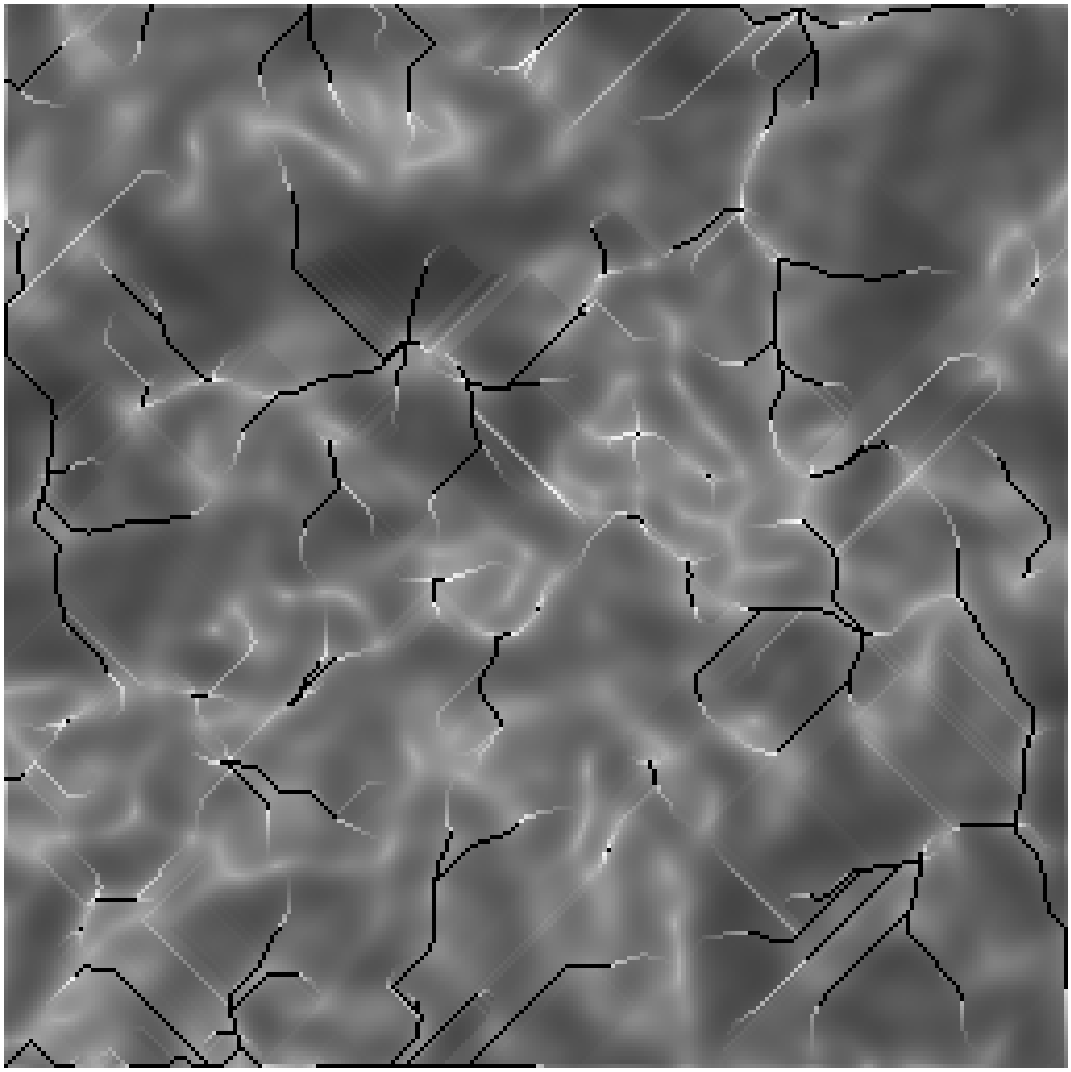

Figure 3: Carte représentant le poids aux différents points de notre terrain

Pour placer nos villes, on choisit simplement les n plus grandes valeurs de la carte. Une fois placées, on va ensuite les faire grossir à l'aide d'un algorithme de modèle d'Eden. A chaque itération, on va sélectionner une ville capable de grandir. Ces dernières peuvent grandir dans 8 directions autour de leur position. On sélectionne une de ces directions et si elle est libre, on agrandit notre ville. Le reste n'est qu'une question de placement. On peut contrôler la taille et l'espacement des villes.

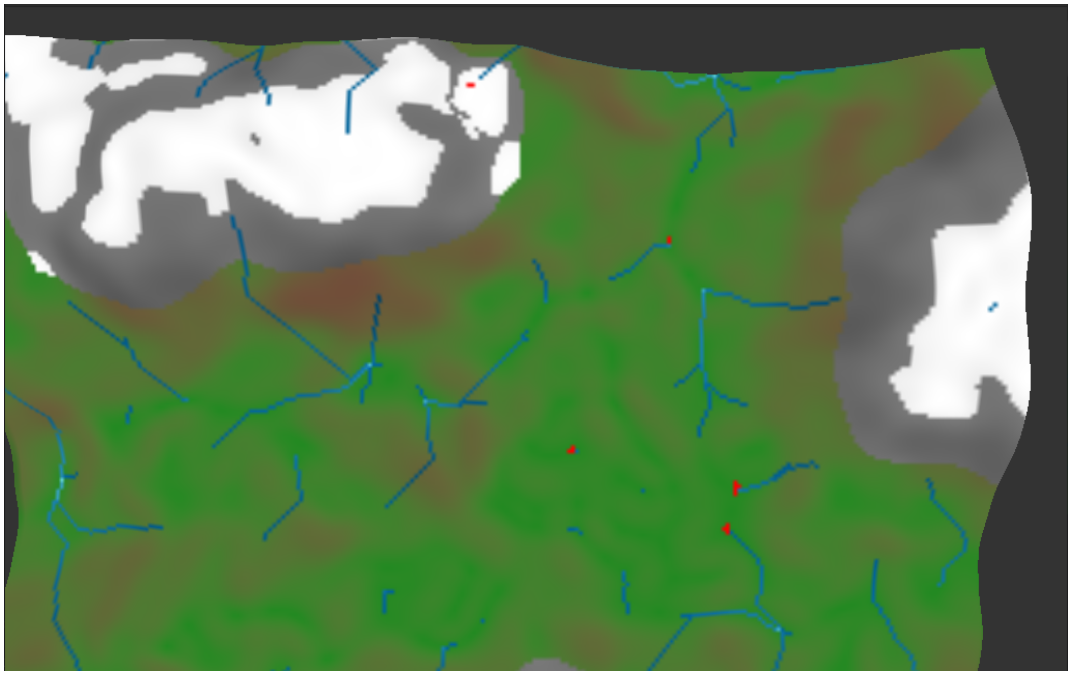

Figure 4: Villes développées grâce à notre algorithme

5 Réseaux routiers

Enfin, nous avons mis en place un algorithme pour relier nos villes entre elles. Pour les routes en elles mêmes, nous utilisons cette implémentation de l'algorithme de Dijkstra. La carte des adjacence est construite à l'initialisation du terrain et est mise à jour au besoin quand le terrain est modifié (pour ne pas avoir à la recalculer à chaque chemin calculé). Le coût pour passer d'un point à un autre est dicté par 3 paramètres: la différence de hauteur, la distance et l'humidité. On cherche à garder un terrain le plus plat possible tout en évitant de s'approcher de zones trop humides.

Ensuite, pour choisir quelles villes relier entre elles, on utilise un algorithme en $O(n^2)$ particulièrement peu efficace mais suffisant pour notre travail. On va parcourir toutes nos villes, et tester toutes les combinaisons possibles pour les relier. Si on a 4 villes a, b, c et d, pour savoir comment relier a et b on va comparer la distance du chemin formé entre a et b avec la distance formé par le chemin ac + cb, puis au chemin ad + db. Si un de ces chemins vérifie l'équation $P + S <= C$, alors on va plutôt choisir de relier a à c puis c à b. Avec p et s les deux sections intermédiaires du chemin, et p le chemin "direct" entre les deux villes.

Ce que cela signifie c'est que si le chemin pour aller de a à b en passant par c n'est pas beaucoup plus coûteux que de faire a à b directement, alors on va préférer ce chemin, pour éviter d'avoir à construire une autre route.

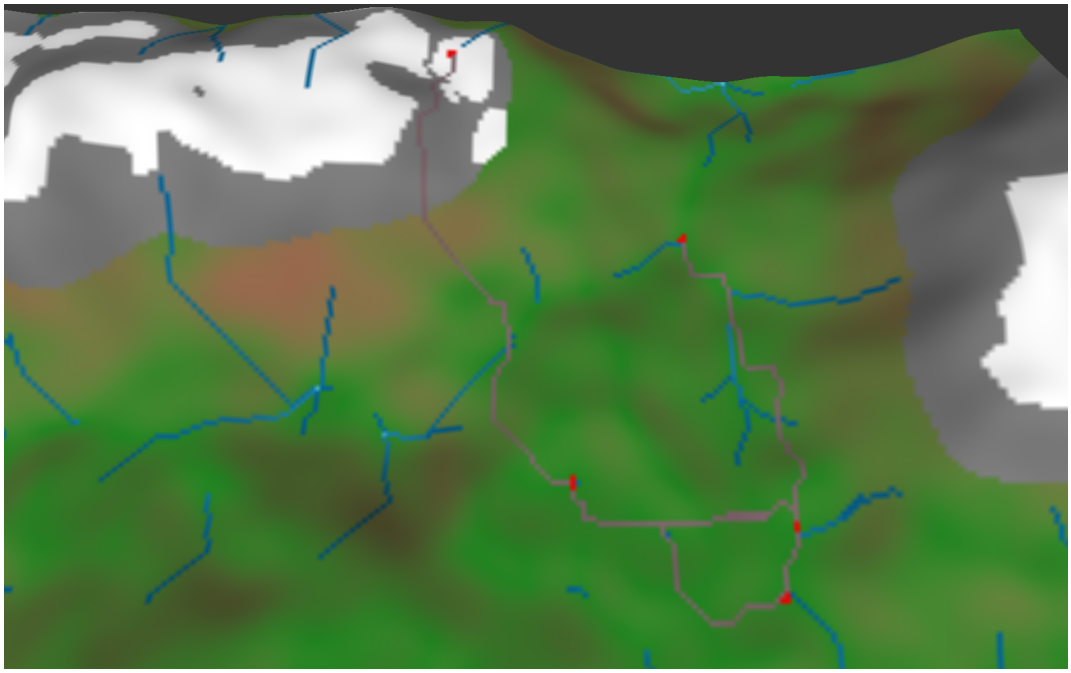

Figure 5: Réseau de routes reliant les villes générées plus tôt

References

[1] Guillaume Cordonnier, Jean Braun, Marie-Paule Cani, Bedrich Benes, Eric Galin, Adrien Peytavie, and Eric Guérin. Large Scale Terrain Generation from Tectonic Uplift and Fluvial Erosion. *Computer Graphics Forum*, 35(2):165–175, May 2016.