Bruit de Perlin et génération de carte

La génération de carte a été un point important de ce projet. En effet, nous cherchions à obtenir une carte aléatoire mais réaliste. Les algorithmes cherchant à atteindre des comportements imprévisibles utilisent trop souvent à tort uniquement des générateurs de nombres aléatoires. Le problème de données complétement aléatoire est qu’elles rendent le résultat peu réaliste et peu crédible. Le bruit de Perlin est une des solutions permettant d’obtenir des résultats aléatoires et naturels.

# Introduction au bruit de Perlin

Une observation de la nature nous donne l’idée directrice ayant mené au bruit de Perlin. En effet, les éléments naturels ont comme point commun leur nature fractale. Un exemple commun peut être l’observation d’une montagne qui, à une large échelle, peut être considérée comme une grande variation de hauteur (la montagne), à une moindre échelle, comme des variations moyennes (les monts), puis de petites variations (rochers), puis de minuscules variations (pierres et cailloux) et ainsi de suite. On peut également remarquer que la fréquence d’apparition des variations est inversement proportionnelle à l’amplitude de la variation. Le bruit final est donc composé d’octaves de fréquence 1/n et d’amplitude n. C’est en partant de ce principe que Ken Perlin a développé ce bruit.

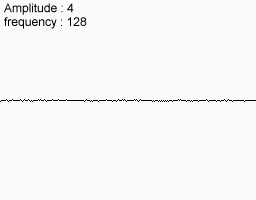
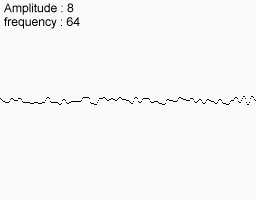
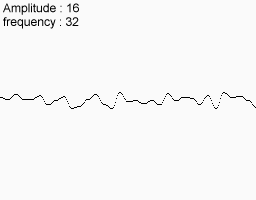
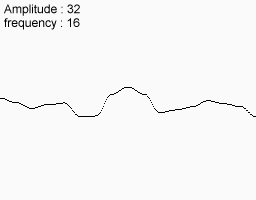
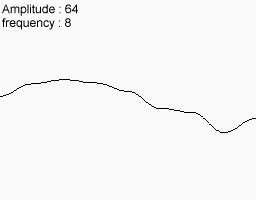
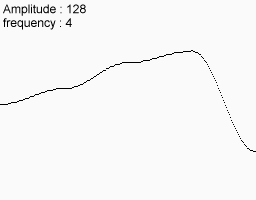


Figure 1 - Fonctions de bruit pour différentes octaves

Le bruit de Perlin nécessite deux éléments principaux : un générateur aléatoire et une fonction d’interpolation. Pour chaque octave, le générateur tirera un nombre de valeurs inverse à la fréquence et dans la plage d’amplitude de l’octave. La fonction d’interpolation sera ensuite utilisée pour obtenir la fonction de bruit associée à l’octave. La génération de bruit de Perlin repose donc sur la génération de fonction de bruit pour des octaves de fréquence différente puis à sommer ces fonctions (Figure 2).

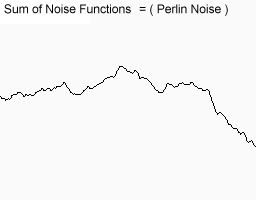


Figure 2 - Somme des octaves

Le principe de ce bruit peut se généraliser à un bruit en N dimensions. Dans notre cas, nous utiliserons une génération en deux dimensions pour la création de la carte.

# Bruit Simplex

Le bruit de Perlin permet d’obtenir des rendus naturels mais le problème de sa complexité () peut poser problème lorsque l’on génère un bruit dans un espace de plus grande dimension. La variante du bruit Simplex permet d’obtenir le même résultat que le bruit de Perlin pour une complexité bien plus faible () pour un espace de plus grande dimension. C’est cette variante que nous utiliserons pour la génération de carte. Le générateur de nombre aléatoire choisi est le Mersenne Twister présent dans la STL. Afin d’optimiser la génération de bruit, nous utiliserons l’implémentation tirant profit de fonctionnalités du C++11 de Solarian Programmer[[1]](#footnote-1).

# Génération de carte

La génération de carte passe par différentes étapes : la création du terrain et la population de celui-ci. Le terrain est créé avec un coefficient d’échelle définissant la fréquence du bruit de Perlin (Figure 3). Un coefficient plus grand permet d’obtenir une carte avec des variations de hauteur plus lentes. La valeur d’élévation à un point donné est obtenue par la récupération de la valeur de bruit aux coordonnées du point (mises à l’échelle). Cette génération permet d’obtenir des altitudes aux points de la carte réalistes.

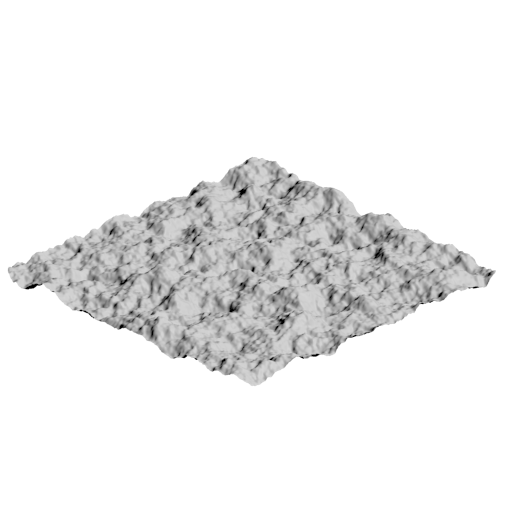
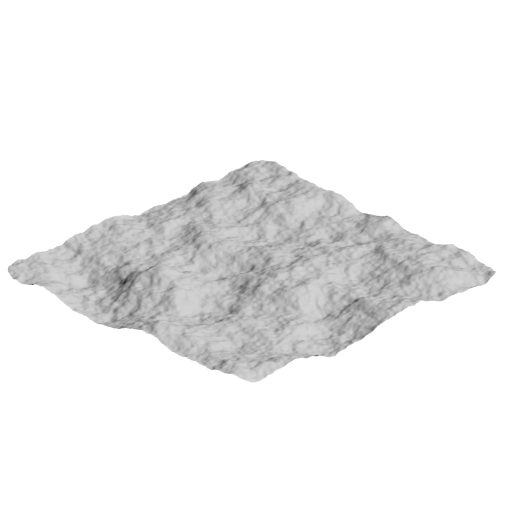


Figure 3 - Génération avec coefficient d'échelle de 0.25 (à gauche) et 0.50 (à droite)

Après cette phase de génération d’élévation, la carte est peuplée par les ressources qui y seront disponibles. Cette phase nécessite des paramètres d’échelle pour le bruit de Perlin et de densité de la ressource. La présence d’une ressource à un point donnée est déterminée par la valeur du bruit à ce point (avec les coordonnées mises à l’échelle). Si cette valeur est comprise dans l’intervalle alors la ressource est affectée. Le Tableau 1 donne les répartitions de ressources générées en fonction des paramètres de densité et d’échelle ( ces évaluations sont effectuées avec le seed 3 pour le bruit de Perlin.

|  |  |  |  |
| --- | --- | --- | --- |
| Densité 0.05 | Densité 0.15 | Densité 0.25 |  |
|  | | | Echelle 20.0 |
| Echelle 5.0 |
| Echelle 1.0 |

Tableau 1 - Répartition des ressources en fonction des paramètres de génération

1. http://solarianprogrammer.com/2012/07/18/perlin-noise-cpp-11/ [↑](#footnote-ref-1)