GENERATION PROCEDURALE
-
PLANETE TELLURIQUES

Algorithmes suivi d'une implémentation en Java.

Yacine Petitprez, Août 2007

Partie 1 : Génération de relief
Partie 2 : Génération des couleur
Partie 3 : Génération de la sphère 3D
Partie 3: Exemple en Java et OpenGL avec JMonkeyEngine [A FAIRE]

Préface

Ce document a été écrit de manière personnelle, pour aucun autre but que celui d'exposer ici ce que je pense être de bonnes méthodes pour générer procéduralement des planètes, dans le cadre par exemple, d'une application temps réel (jeu vidéo etc…).
De ce fait, ne cherchez pas l'exactitude physique de l'ensemble des algorithmes défini ci-dessous !

Je suis ouvert a toute remarques, propositions, améliorations ou encore critiques, que vous pouvez me faire parvenir à cette adresse : *anykeyh@gmail.com*

Bonne lecture !

Introduction

La génération de terrain est un très vaste sujet demandant quelques connaissances dans les techniques fractales.
La génération d'une planète entière par le biais de techniques procédurales demande une approche sensiblement différente.
En effet, celle-ci fait appel à de nombreux paramètres plus ou moins complexes à implémenter :

La température du terrain est modulée par le cosinus de l'angle du rayon d'incidence de la lumière vers le sol.

La mise en place de la coloration et de la hauteur du terrain de planète est totalement différente en fonction de la densité d'atmosphère ou encore des composantes ou de la température du sol.

La lune par exemple possède un terrain lié à son exposition aux météorites, tandis que la terre a vu le sien renouvelé par la tectonique des plaques et l'érosion naturelle.

La création de continents est régie par des lois non uniforme de distribution des hauteurs : Il y a des zones océaniques, et des zones continentales.

Ainsi, on peu intuitivement extraire plusieurs paramètres utiles à la conception d'un terrain de planète:

- Température à un point du terrain.
- Pente à un point du terrain
- Hauteur du terrain
- Densité de l'atmosphère

Le modèle, devant être utilisé dans la génération temps réel, est très approximatif. Ainsi, il a été décidé de ne pas prendre en compte l'obliquité de l'astre (pas de gestion des saisons), ou le facteur d'humidité.

De plus, divers modèles ont été mis en place pour réaliser les différents type de planète : La densité d'atmosphère fait office de seuil pour le choix de l'algorithme de génération des hauteurs à mettre en place :
Une atmosphère de faible densité signifiera l'usage d'une carte des hauteurs basé sur la modification par impact de météores (« cratérisation »), une atmosphère à haute densité utilisera un autre algorithme cité plus bas…

Nous verrons donc dans un premier temps comment générer un relief assez proche de la réalité. Nous verrons ensuite comment peindre ce relief de façon réaliste, et réaliseront une implémentation en Java de la génération à l'affichage sur une sphère 3D avec les dernières techniques de rendu (shaders).

1ere partie : Génération de relief

Quelques remarques

Il est très difficile de créer une texture rectangulaire pour un monde sphèrique, en conservant hauteur et surface de manière uniforme.
Ainsi, plusieurs techniques de projections peuvent être utilisé pour générer la texture d'une sphère.
Dans ce document, nous nous affranchirons de la notion de sphère, préférant générer une texture comme sur un cylindre (projection de Mercator).
De ce fait, les points présent prés des pôles risquent de subir une compression, notamment sur les sphère (moins sur les sphères de type géode).
De plus, nos textures devront gérer la rotation des coordonnées, afin d'éviter la présence d'une coupure entre l'extrême droite et l'extrême gauche.
Pour l'horizontale, un simple modulo par la taille de la texture permet de boucler le terrain.
Pour les coordonnées verticales, c'est légèrement plus compliqué. Lorsqu'un observateur descend au pole Sud, il ne remonte effectivement pas au pole Nord. Seul la longitude change, s'inversant.
De ce fait, on trouve une relation de ce type :

$$p_{x,y} = \begin{cases} \left| x + \frac{l}{2} \right| & \text{si } y < 0 \\ -y & \\ \left| x + \frac{l}{2} \right| & \text{si } y \geq h \\ 2h - y & \end{cases}$$

Terrain pour les planetes de type terrestre

La première chose qui vient à l'esprit lors du développement d'un terrain numérique est la carte du relief, encore appelée matrice numérique de terrain (MNT) ou « *height map* ».
Celle-ci utilise un bitmap, généralement en niveau de gris sur 1 octet (soit 256 valeurs possibles), où la hauteur d'un point est proportionnelle à la valeur de chacun des pixels.

Il existe diverse façon de générer procéduralement une carte des hauteurs.
Un technique simple et efficace est la génération par bruit fractal, ou *Perlin noise[1]*.
Cette technique s'apparente à sommer plusieurs couches de bruit de différentes fréquences (*fig.* 1).

$$p_{x,y} = \sum_{i=1}^{k} \frac{f_i(x,y)}{2^i}$$

Fig. 1) Algorithme « Perlin noise ». f_i représente une fonction de bruit de fréquence 2^i

Généralement, les implémentations font usage de couches de bruit (ou octaves) de plus en plus petites, qui sont interpolées puis mixée entre elle, formant la texture finale.
Le *Perlin noise* offre ainsi un rendu fractal à un faible coût de calcul.

Cependant, ce rendu est uniformément distribué. Or, sur la Terre par exemple, la hauteur est distribuée de façon moins uniforme, plus regroupé sous forme de continents.
Ceci est du à la tectonique des plaques. Sans entrer dans les détails, les plaques océaniques, plus lourdes, s'enfonce dans le manteau terrestre plus facilement que les plaques continentales. Il en résulte un contraste de quelques kilomètres entre la hauteur d'une plaque continentale et celle d'une plaque océanique.

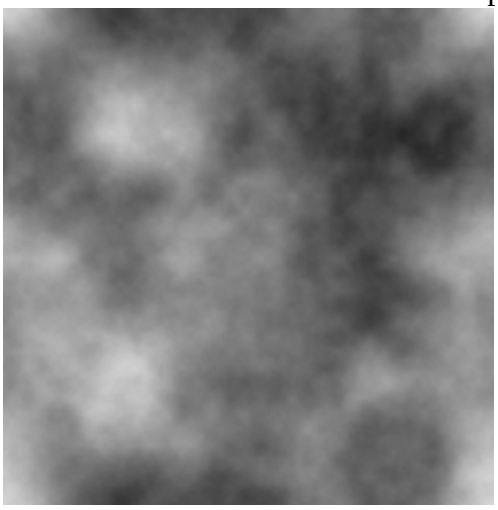


Fig. 2) Perlin noise avec Kmax = 7

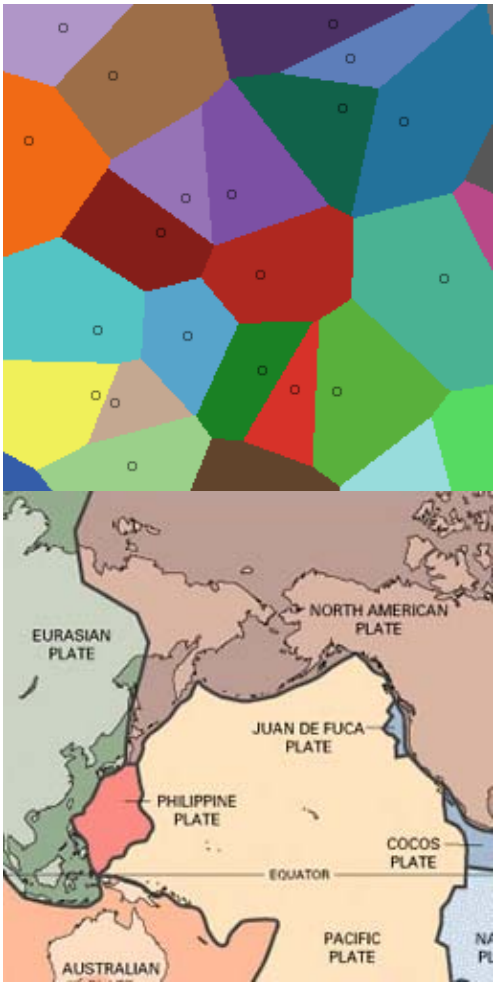
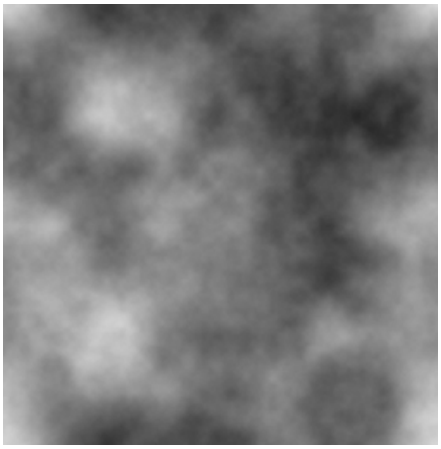
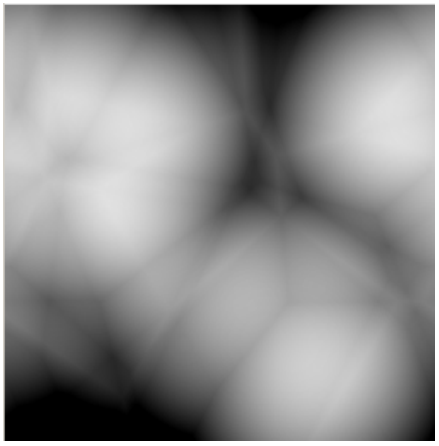


Fig. 4) Diagramme de Voronoï et structure des plaques tectoniques

Le but ici n'est pas de développer un modèle de plaques tectoniques, mais de produire un relief qui privilégiera les structures en continent.
De ce fait, on remarque de suite la similitude entre un diagramme de Voronoï et l'aspect des plaques (fig. 4)



X



=

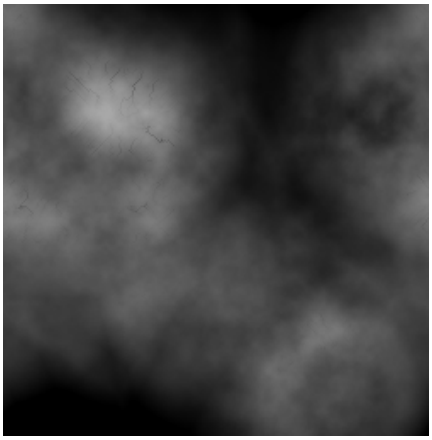


Fig. 5) Modulation de la carte (fig. 2) par le diagramme de coefficients a = {-1, -2, 1.5}. Les rivières ont aussi été rajouté (visible en haut a droite).

On se penche donc sur l'utilisation d'un diagramme de Voronoï modifié[2] pour moduler la carte précédente.
Pour chaque pixel, on tri les centres du diagramme par leur distance relative au pixel. On calcul alors la valeur du pixel équivalent comme étant la somme des distance multipliés par un coefficient a_n qu'on choisi en fonction de l'aspect de notre structure.

$$p_{x,y} = \sum_{i=0}^{n} a_i D_i$$

Fig. 3) Diagramme de Voronoï modifié. a_n représente un coefficient et D_n représente sa distance au n^{eme} centre le plus prés du pixel.

Grâce à cette technique (fig. 2), les zones claires représentent les zones de plaques continentales, tandis que les zones sombres représentent les zones océaniques, où le niveau est beaucoup plus bas.
Ainsi, les structures cellulaires permettent de former de large zone d'altitude ou de profondeur et ainsi de casser la distribution uniforme du bruit fractal.

Ajout de rivières

Pour finaliser, il est possible d'ajouter des cours d'eau. Ceci renforcera nettement l'aspect des cotes, créant par la même des côtes proches des fjords présents en Europe du nord.
Pour cela, nous prenons un point immergé de la carte au hasard. En ce point, nous cherchons le point le plus bas parmi ses voisins.
Nous décrémentons d'une certaine valeur la hauteur du point courant, et allons sur le point voisin sélectionné.

```
plusPetitVoisinNonVisité(x, y) : Point2D
Var
      Retour : Point2D
      I, J : Entier
Début
      Retour.x := x – 1
      Retour.y := y – 1
```

```
      Pour  I de -1 à 1 Faire
            Pour J de -1 à 1 Faire
                  Si  P[X+I,  Y+J]  <  P[Retour.x,  Retour.y]  ET  Non  P[X+I,
Y+J].visité Alors
                        Retour.x = X+I
                        Retour.y = Y + J
      Fin si
            Fin pour
      Fin pour

      Retourner Retour
Fin

ajoutRiviere(x, y)
Var
      Nouveau : Point2D
Début
      Visité[x, y] = VRAI
      P[x, y] = P[x, y] – 1

      Nouveau = plusPetitVoisinNonVisité(x, y)
      ajoutRiviere(Nouveau.x, Nouveau.y)
Fin
```

Fig. n) Ajout de riviere

Conclusion

L'implémentation actuelle mets environ 5 secondes pour créer une carte 512*1024 sur AMD Athlon 3000+.
Cependant, l'implémentation de la couche de Voronoï pourrait être grandement optimisé. Actuellement, un tri est effectué sur chaque pixel afin de déterminer l'ordre des points!

Terrain pour les planètes de type lunaire : cratérisation

PARTIE À FAIRE !

Partie 2 : Génération de la carte de couleurs

La carte des couleurs, permet d'afficher les couleurs réelles du terrain, tel qu'on le verrait depuis l'espace. Dans le cas d'une planète habitable, cela reviendrait à représenter un terrain a dominance bleu (l'eau), avec des plaines vertes (herbe & foret), des calotte glaciaire blanche etc…

Divers algorithme sont disponible, le plus simple étant de créer une palette de couleur en fonction de la hauteur de chacun des points du terrain.

Dans l'algorithme suivant, nous allons utiliser la distributivité selon des lois normales pour créer un terrain digne de celui de la Terre.
On part du principe que chaque parcelle minuscule (non discrétisé ici) de terrain [x, y] peut être du type de terrain T ou non. On retrouve une probabilité binaire, en fonction de divers paramètres, tels la hauteur, la température ou encore la pente.

Dans le cadre d'une carte du monde, chaque pixel constitue à lui seul des centaines de kilomètres carrés de terrain. De ce fait, on peu dire que le nombre de parcelle unitaire de terrain tend vers l'infini, et que la moyenne de l'ensemble des tirages de points (X) est égale à la probabilité P(X).

$$\lim_{n \to \infty} \sum^{n} \frac{X}{n} = P(X)$$

Approximation de la moyenne de n tirage d'une variable aléatoire X binaire (1 ou 0)

On admet que chaque type de terrain suit une loi pseudo normale d'optimum température = T, pente = P et hauteur = H. On y ajoute l'écart-type pour ce type de terrain, respectivement vT, vP, vH.

$$P_{x,y}(X) = e^{-\left(\frac{|T-t|}{vT} + \frac{|P-p|}{vP} + \frac{|H-h|}{vH}\right)}$$

Loi pseudo normale pour un terrain
Probabilité que X vaut 1.
t, p et h sont les valeur température, pente et hauteur du terrain au pixel x,y de la carte

À noter que cette loi est « pseudo normale », dans la mesure où elle ne suit pas tout a fait la loi normale (l'exposant devrait être monté au carré !).
Ceci est nécessaire, de par la mauvaise précision des flottants.

Par exemple, on peut dire qu'un terrain d'herbe suit une loi pseudo normale de paramètres T = 18°, dT = 3.5, P = 0, dP = 2, H = 0 dH = 3.
En d'autre terme, l'herbe a comme milieu naturelle une température de 18°, une pente faible ou nulle, et une hauteur au niveau de la mer.

$$p = max(|P_{x,y} - P_{x-1,y}|, |P_{x,y} - P_{x+1,y}|, |P_{x,y} - P_{x,y+1}|, |P_{x,y} - P_{x-1,y}|)$$

$$t = \cos\left(\frac{\left|x - \frac{h_{max}}{2}\right|}{h_{max}}\pi\right)(T_{eq} - T_{pol}) + T_{pol} - H_f \, max(0, P_{x,y} - W_{level})$$

$$h = P_{x,y}$$

Calcul des attributs p, t et h. P[x, y] est la valeur de la hauteur du relief, T_{eq} est la température à l'équateur, T_{pol} est la température aux pôles, H_f est un facteur variable, W_{level} est la hauteur de l'eau.

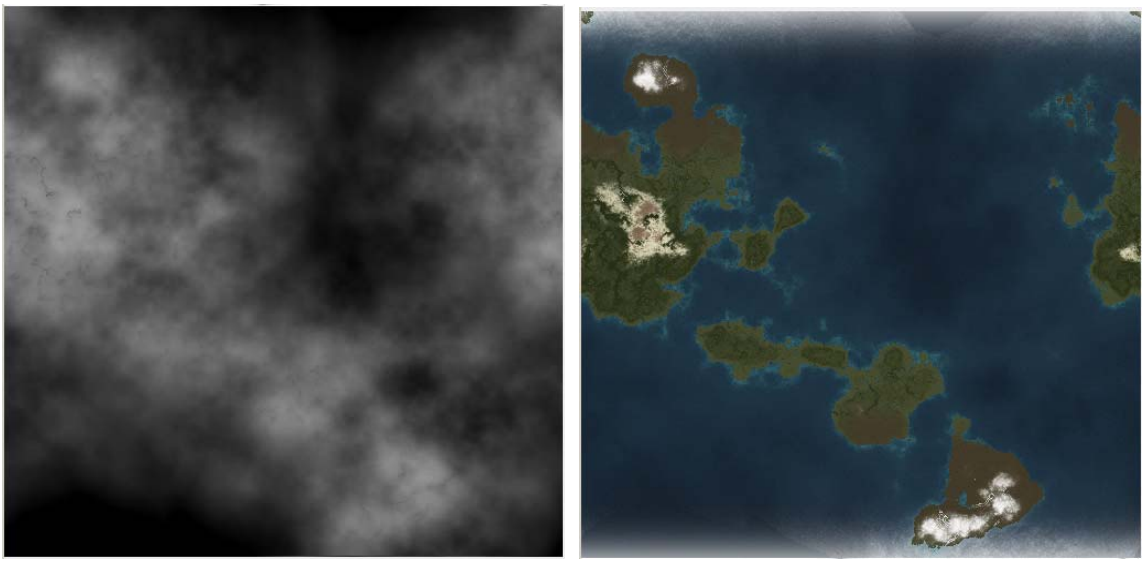
Cependant, la somme de l'ensemble des probabilités n'est pas forcement égal à 1.
On pondère donc la couleur de chaque type de terrain par sa probabilité, et divisons par la somme des coefficients.

$$p_{x,y} = \sum_{j=0}^{n} \frac{P_j(X)_{x,y} C_j}{\sum_{k=0}^{n} P_k(X)_{x,y}}$$

Etape de mixage. p[x, y] est la couleur au point [x,y] de la carte des couleurs. C[j] représente la couleur du type de terrain numéro j

J'y ajoute une fonction de seuillage, ou le coefficient vaut zéro (absolument pas de ce terrain) notamment de la hauteur. Ainsi l'eau est seulement présente lorsque $h <= W_{level}$, et inversement pour l'herbe.

En utilisant l'exemple de palette en annexe, voici le résultat :



Le relief généré, et sa version colorée

Exemple :

Paramètres (TODO : annexes + format xml) :

Température à l'équateur : 50°
Température aux pôles : -20°
Niveau de l'eau : 100
Hf = 0.1

Palette de terrain employé :

Tundra: Hmin = Wlevel, H = Wlevel, vH = 4
P = 0, vP = 2.5
T = 0, vT = 1.5
Couleur = 0x4113A28 (RGB)

Herbe: Hmin = Wlevel, H = Wlevel, vH = 3
P = 0, vP = 2
T = 18, vT = 3.5
Couleur = 0x474D30

Foret équatoriale: Hmin = Wlevel, H = 1.1*Wlevel, vH = 3
P = 5, vP = 6
T = 25, vT = 2
Couleur = 0x272F16

Eau très profonde: Hmax = Wlevel, H = 0, vH = 50
P = 0, vP = 5
T = 50, vT = 5
Couleur = 0x101924

Eau profonde : Hmax = Wlevel, H = 0.75 * Wlevel, vH = 50
P = 0, vP = 5
T = 50, vT = 5
Couleur = 0x203849

Eau peu profonde : Hmax = Wlevel, H = Wlevel, vH = 5
P = 0, vP = 1.5
T = 50, vT = 5
Couleur = 0x4FAAA8

Neige : T = -100, vT = 6 (Les autres coefficients sont désactivés)
Couleur = 0xFFFFFF

Desert:
Hmin = Wlevel, H = Wlevel*1.3, vH = 5
P = 0, vP = 0.25
T = 50, vT = 3
Couleur = 0xE5DFAD

Montagne:

Hmin = Wlevel, H = Wlevel * 1.6, vH = 2
P = 100, vP = 10
T = 10, vT = 2

Partie 3 : Génération 3D

Génération de l'atmosphère

Le passage d'un rayon de lumière dans une atmosphère crée deux effets physiques :

- La diffusion : Une partie de l'énergie lumineuse se diffuse
- Absorption de lumière : La lumière passant dans l'atmosphère subit une absorption relative à un spectre d'absorption.
  Pour la Terre, l'absorption est de type bleuté, et les rayons passant dans l'atmosphère subissent un décalage vers le rouge proportionnel à la distance parcourue dans l'atmosphère, d'où la couleur rouge des couchés de soleil.

La diffusion peut être approximé par cette relation :

$$C_{out} = C_{atm}(1 - \vec{n}.\vec{v})(\vec{n}.\vec{l})$$

$\boldsymbol{C_{out}}$ représente la couleur du pixel en sortie, $\boldsymbol{C_{atm}}$ représente la couleur de l'atmosphère, ***n*** est le vecteur normal au vertex courant, ***v*** le vecteur direction vers la vue et ***l*** le vecteur direction vers la source de lumière.

Note : ***l***, ***n*** et ***v*** doivent être normalisés. Le « . » représente le produit scalaire.

Le résultat en 3D

Avec et sans atmosphere

Carte précédente, sur une sphère 3D, rendu final. Les nuages ne sont par contre pas générés procéduralement. (C’est pour plus tard ;-))

Références :

[1] http://freespace.virgin.net/hugo.elias/models/m_perlin.htm
[2] http://oddlabs.com/download/terrain_generation.pdf, pages 3-5