

Modélisation procédurale de terrain avec végétation et routes

Par Julien Fleckinger et Cassandra Breton

Plan

[But du projet 2](#_Toc505341654)

[Classes de bases 2](#_Toc505341655)

[La pente 2](#_Toc505341656)

[Le drainage 3](#_Toc505341657)

[Le wetness index 3](#_Toc505341658)

[Le power stream 4](#_Toc505341659)

[La végétation 4](#_Toc505341660)

[Les routes 6](#_Toc505341661)

[Résultats 6](#_Toc505341662)

## But du projet

Le projet consistait à générer un terrain en travaillant sur 2 des 3 sujets qui étaient l’érosion, la végétation et les routes. Nous avons choisi de travailler sur la végétation et les routes.

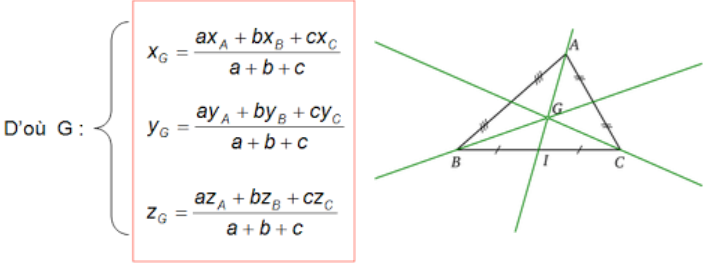
//TODO mettre image du rendu 3D avec routes et arbres

## Classes de bases

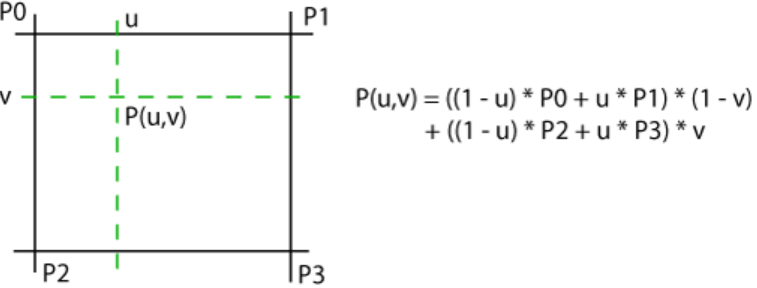
Tout d’abord nous avons créé les classes Box (boite englobante) et Array (permettant de créer une grille) afin de pouvoir créer notre classe ScalarField. Nous avons aussi fait une classe image (qui charge ou génère des images), Maillage (gestion de maillage), Render (pour le rendu 3D), Vector (pour tous ce qui est gestion de points) et Terrain (regroupant tous les résultats des différents calculs).

Notre classe ScalarField nous permet de calculer des gradients, d’interpoler avec les méthode triangulaire et bilinéaire, de lire des images et de normaliser des valeurs.

Interpolation triangulaire :



Interpolation bilinéaire :



La classe HeightField a été implémenté sur la base de la classe ScalarField afin de pouvoir créer le maillage résultat, de pouvoir modifier la taille de la grille, mais surtout de pouvoir calculer la pente, le drainage, la wetness index et le power stream.

## La pente

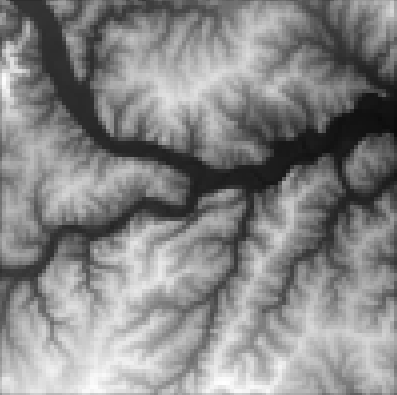
 

Figure 1 – Terrain de base Figure 2 – Pente calculée à partir

du terrain donné

Afin de calculer la pente on calcule simplement la norme du gradient en chaque point de notre terrain en prenant garde aux points se trouvant sur des extrémités.

Calcul de gradient :



## Le drainage

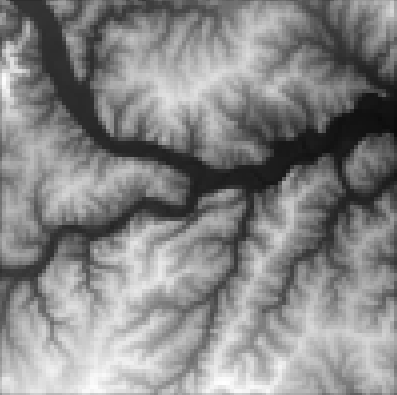
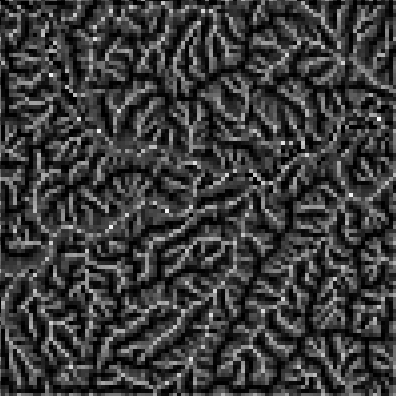
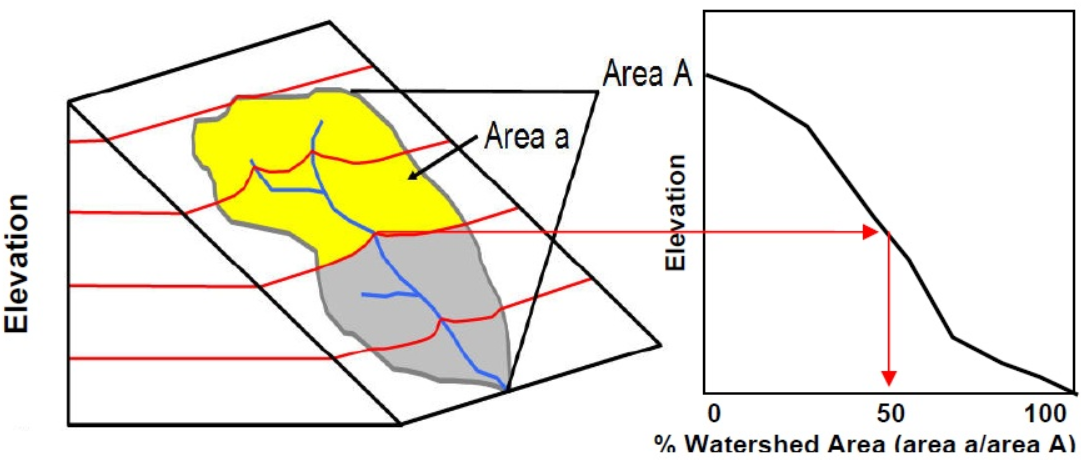
 

Figure 3 – Terrain de base Figure 4 – Drainage calculée à partir

du terrain donné

Le drainage est calculé en fonction d’une valeur d’écoulement et d’une position. Ensuite, on distribue cette valeur afin de savoir où ira l’eau.



## Le wetness index

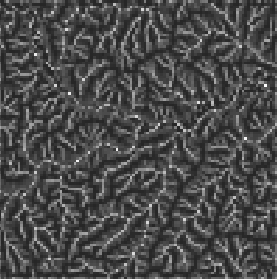
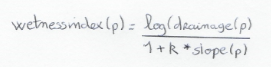
  

Figure 5 – Pente Figure 6 – Drainage Figure 7 - Wetness index calculé

Le wetness index est calculé grâce au ScalarField contenant les valeurs de pentes et celui contenant les valeurs de drainage et une valeur k.

On fait donc pour chaque point :



## Le power stream

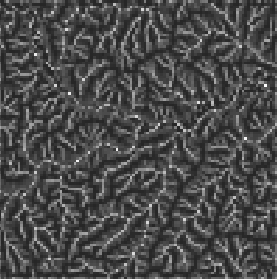
  

Figure 8 – Pente Figure 9 – Drainage Figure 10 - Streampower calculé

Tout comme le wetness index, on a besoin du ScalarField contenant les valeurs de pentes et celui contenant les valeurs de drainage.

Ensuite, on fait pour chaque point :



## La végétation

Notre simulation de végétation nécessite le ScalarField contenant la pente, celui contenant la hauteur et celui avec le wetness index.

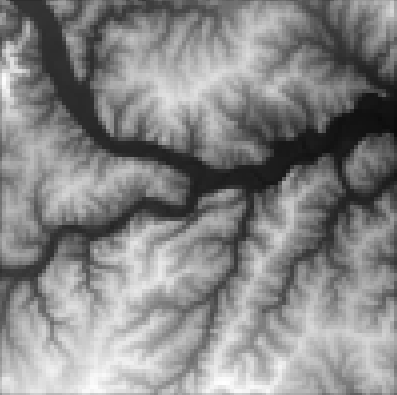
  

Figure 11 – Pente Figure 12 – Hauteur Figure 13 - Wetness index



Figure 14 – Végétation calculée à des hauteurs,

des pentes et du wetness index

Les classes crées pour la simulation de végétation sont Veget (dans laquelle est les données de l’arbre sont stockée : l’âge, l’espérance de vie, la position, la canopée, la valeur de développement, celle de maturité, la hauteur, le type de l’arbre et toutes les valeurs de contraintes d’existence, telles que les valeurs min et max de wetness, de pente, d’altitude, de taille et de canopée) et Foret (qui stocke les arbres, la densité de végétation et permet de lancer la simulation).

Nous avons deux types d’arbres : des sapins et des pommiers. Leurs données sont :

Sapin :

* Slope max = 0,7 (radians)
* Altitude min = 0 m
* Altitude max = 2000 m
* Wetness min = 0,1
* Wetness max = 2
* Hauteur max = 60 m
* Canopée max = 7
* Esperance de vie = 500 ans
* Maturité = 20 ans

Pommier :

* Slope max = 0,5 (radians)
* Altitude min = 0 m
* Altitude max = 1000 m
* Wetness min = 0,05
* Wetness max = 2
* Hauteur max = 10 m
* Canopée max = 10
* Esperance de vie = 100 ans
* Maturité = 10 ans

Notre génération de forêt débute avec un nombre de naissance et de mort à 0.

On commence alors par regarder pour chaque arbre s’il doit mourir, pour ce faire, on regarde si son âge est supérieur à la moitié de son espérance de vie. Dans ce cas, on lance un jet de probabilité qu’il meurt et si tel n’est pas le cas, on diminue sa chance de survivre la prochaine fois. Ensuite, s’il est assez développé on l’agrandie et on met à jour sa canopée.

Par la suite, on cherche maintenant à savoir s’il y a eu des naissances, pour ce faire, si l’arbre est à maturité, on lance un aléatoire pour savoir s’il y a une naissance. Et si tel est le cas, on ajoute un arbre.

On doit maintenant placer ces nouveaux arbres en fonction de la densité de végétation et de leur affinité en termes de valeurs de pente, wetness index et hauteur aux positions visées par ces naissances.

On peut donc boucler sur un certain nombre d’années pour voir l’évolution de la région.

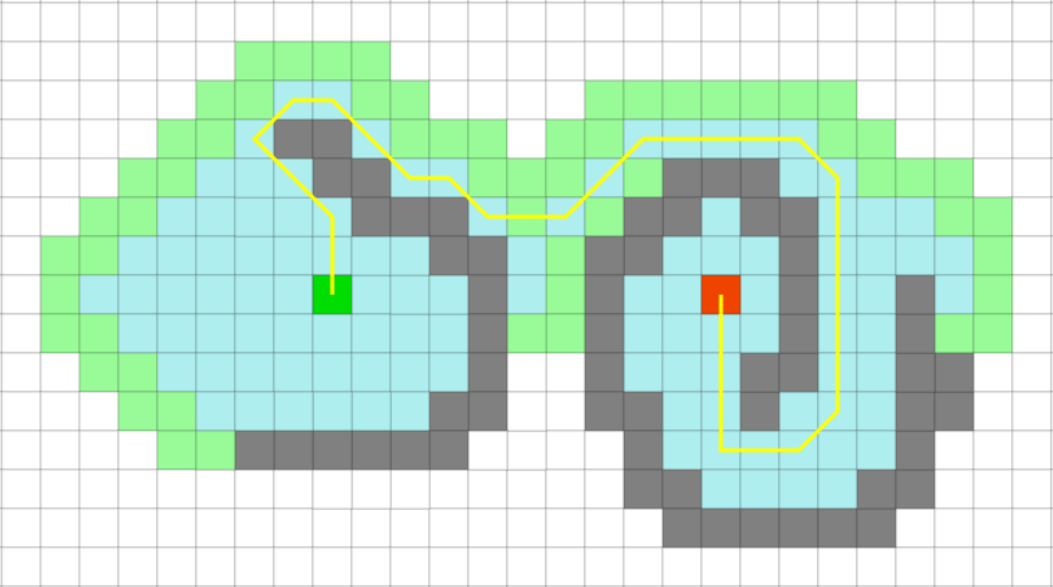
## Les routes

Figure 15 – Pente calculée Figure 16 – Route calculée

Pour la génération de routes, on s’est basée sur la pente et sur A\*. Pour ce faire, nous avons créés deux classes : Case (contenant les informations nécessaires en chaque point pour le calcul de l’heuristique et du cout dans A\*) et Route (qui lance A\* sur le ScalarField de pente du terrain et qui donnera le chemin résultat et le cout de celui-ci).

Schéma (sans prise en compte de la hauteur de A\*) :



Pour ce faire, on donne à A\* nos deux positions (départ et objectif) et on applique un simple A\* avec comme heuristique la distance et la valeur de pente (les valeurs de pentes trop extrêmes auront comme cout la valeur INFINITY).

## Résultats

//Images de rendu 3D avec et sans noise