Projet 2 : Analyse et Synthèse de Terrains

Ce projet est effectué en C++. Plusieurs types de données fondamentaux seront à définir, comme les vecteurs dans R^2 et R^3 *Vec2* et *Vec3* avec les surcharges d'opérateurs nécessaires.

Box2

Définir une classe *Box2* dans le plan, elle servira à définir des domaines rectangulaires dans R^2. Ecrire ses fonctions membres : *Inside*, *Intersect*, permettant de tester si un *Vec2* est dans ou hors une *Box2*, et calculant si deux *Box2* se coupent.

Grille

Pour traiter des cartes de scalaires, il est nécessaire d'effectuer des traitements sur une grille régulière. Définir une classe *Grid*, dérivant de *Box2*, caractérisée par sa discrétisation et implémentant plusieurs fonctions sur les coordonnées entières dont *Index* permettant de trouver l'index d'un point (i, j) dans le tableau, *Inside* testant si (i, j) est dans les bornes.

Champs scalaires

Pour traiter de manière générique les champs de hauteurs et les cartes d'analyse, définir une classe *ScalarField* dérivant de *Grid* et stockant le tableau de valeurs scalaires. A l'aide de Qt, écrire une fonction permettant de construire et de sauver un *ScalarField* sous la forme d'une image en couleur ou en niveau de gris.

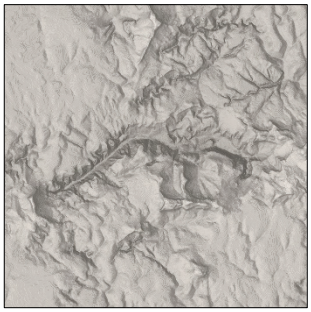
Analyse et traitements

Ecrire les fonctions *Gradient* calculant le gradient en un point (i, j) et les fonctions construisant les *ScalarField* de norme du gradient *GradientNorm* et de laplacien *Laplacian*. On pourra également coder différent lissages avec des masques 3×3 par exemple *Smooth* (masque en 1,2,4) et *Blur* (masque avec des 1).

Coder les fonctions de normalisation *Normalize* et de seuillage *Clamp*.

Cartes de hauteur

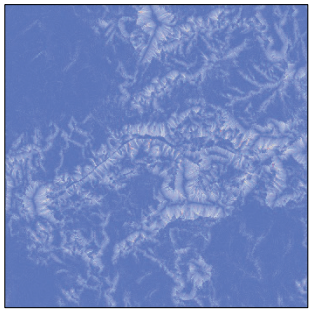
Une carte de hauteur peut être considérée comme un *ScalarField* particulier : créer une classe dérivée *HeightField* avec ses constructeurs et des fonctions membres de calcul de hauteur *Height*, de pente *Slope*, de pente moyenne dans les huit direction *AverageSlope*, pour un couple (i, j).



Pour pouvoir générer facilement un maillage, écrire les fonctions de calcul d'un point $\mathbf{p}_{ij}$ *Vertex* et de sa normale $\mathbf{n}_{ij}$ *Normal* pour (i, j). A partir cette dernière, écrire une fonction *Shade* générant une image de relief en fonction d'une direction d'éclairement. Ecrire une fonction *Export* exportant la carte au format OBJ.

Eric Galin
eric.galin@liris.cnrs.fr

Analyse



Ecrire des fonctions d'analyse d'un *HeightField* calculant des *ScalarField* soit directement dans la classe (algorithme spécifique aux champs de hauteur) soit dans la classe mère : *StreamArea* pour le calcul de l'aire du drainage A, *StreamPower* définit comme $P = A\, s^{1/2}$, *Slope* calculant la pente. On pourra aussi étudier d'autres caractéristiques comme le *WetnessIndex* défini comme $W = \ln(A)\,/s$ (on pourra modifier légèrement la définition en $W = \ln(A)\,/(s+\varepsilon)$ pour éviter les infinis), ou une approximation de l'éclairement global par calcul d'accessibilité *Access* (requiert l'implémentation d'une fonction *Intersect* avec un rayon).

Algorithmes avancés

Une implémentation des algorithmes pour la cohérence d'écoulement (*depression filling* ou *breaching*) est proposée sur **RichDem** : https://github.com/r-barnes/richdem

Erosion

Dans la classe *HeightField* implémenter les algorithmes d'érosion thermique, éventuellement hydraulique, vus en cours. A l'aide des fonctions précédentes, ou pourra coder rapidement les érosions de type Stream Power $\partial z/\partial t = u - k\, A^{1/2}\, s$ avec *StreamArea* et Hill Slope $\partial z/\partial t = -k\Delta z$ avec *Laplacian*.

Rendu

Le rendu sera sous la forme d'un document PDF de deux pages maximum résumant les éléments développés. Il contiendra le nom, prénom, numéro d'étudiant, un pointeur vers le code C++, des captures d'écran présentant les différents éléments, les statistiques (temps de calcul, traitements effectués).

Eric Galin
eric.galin@liris.cnrs.fr