

# [HAI809I - Projet] Compte Rendu N°5



Yahnis Saint-Val ([yahnis.saint-val@etu.umontpellier.fr](mailto:yahnis.saint-val@etu.umontpellier.fr))

Léa Serrano ([lea.serrano@etu.umontpellier.fr](mailto:lea.serrano@etu.umontpellier.fr))

25 mars 2023

Génération procédurale de cartes d'environnement

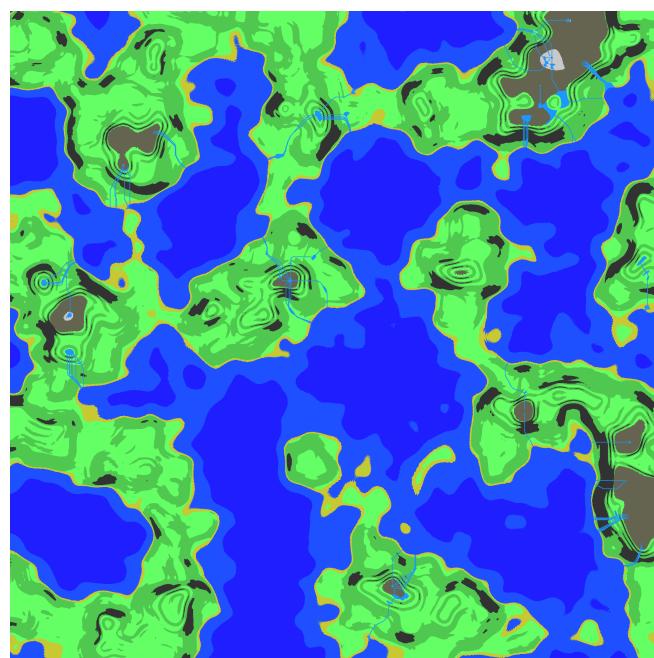


FIGURE 1



# 1 Amélioration de la carte d'altitude

Nous avons amélioré nos cartes d'altitude en générant deux cartes d'altitudes, une normale comme celle qu'on utilisait avant et une deuxième avec beaucoup plus de bruit. Ensuite nous avons mélangé ces deux cartes avec un coefficient de distorsion.

Pour faire cela, nous avons récupéré chaque valeur de la carte 1 tel que  $\text{valueLarge} = \text{valueCarte1}/255$ , et chaque valeur de la carte 2 tel que  $\text{valueSmall} = \text{valueCarte2}/255$ . Ensuite nous avons donné comme valeur à notre nouvelle carte pour chaque pixel :  $(\text{valueLarge} + (\text{valueSmall} - 0.5) * \text{distorsion}) * 255$ .

Cela nous permet d'avoir une carte beaucoup plus nuancée et ça donne un effet plus joli.

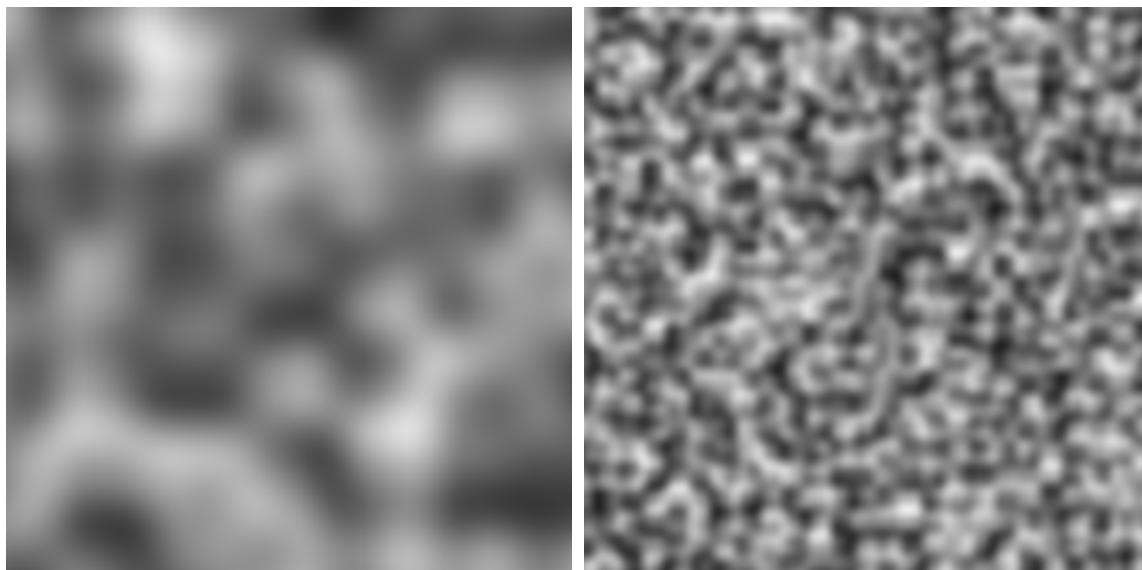


FIGURE 2 – 1ère carte (à droite) et 2ème carte (à gauche)



## Compte Rendu N°5

---

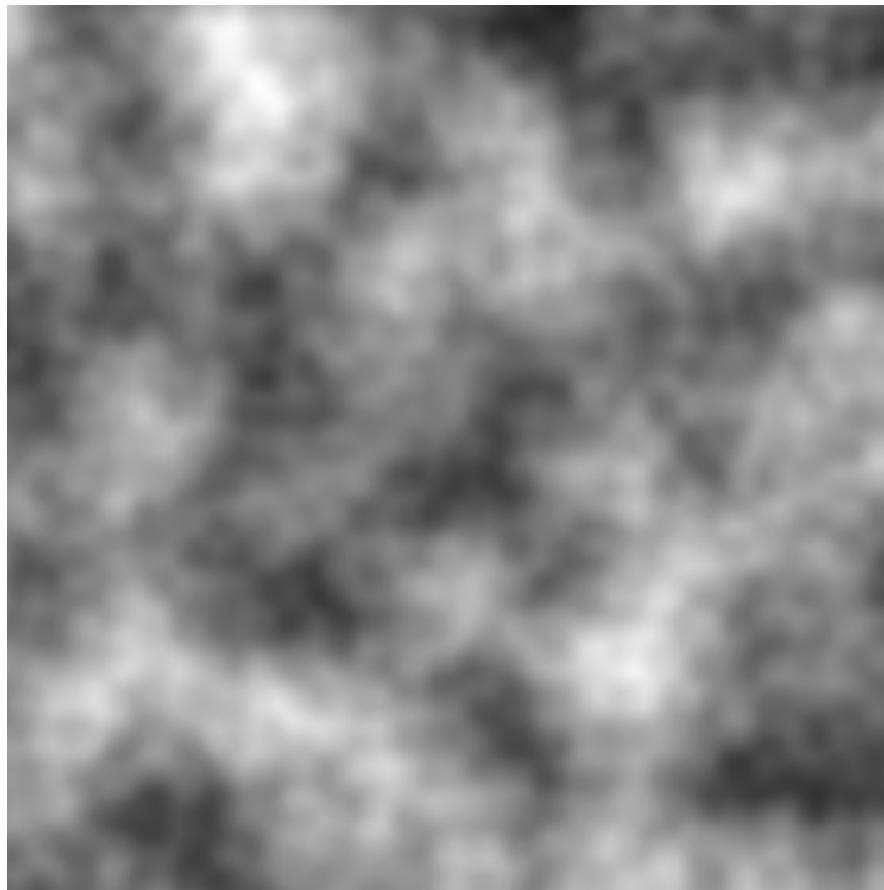


FIGURE 3 – Carte d'altitude obtenue



## 2 Amélioration de la carte de gradient

Nous avons rencontrés quelques problèmes avec la carte de gradients (notamment des valeurs incohérentes à certains endroits) lorsque nous avons essayé d'implémenter la carte de rivières.

Nous avons donc modifié l'algorithme : il est maintenant adaptatif à la taille de la carte (gabarit bi-cubique variable), et on utilise donc les voisins dans "toutes les directions".

Aussi, la direction de la pente ( $dx$  et  $dy$  sur les canaux R et G) sont maintenant normalisés, pour s'assurer d'obtenir des valeurs cohérentes ; l'intensité de la pente (norme du vecteur  $dx, dy$ ) est donc stocké uniquement sur le canal B.

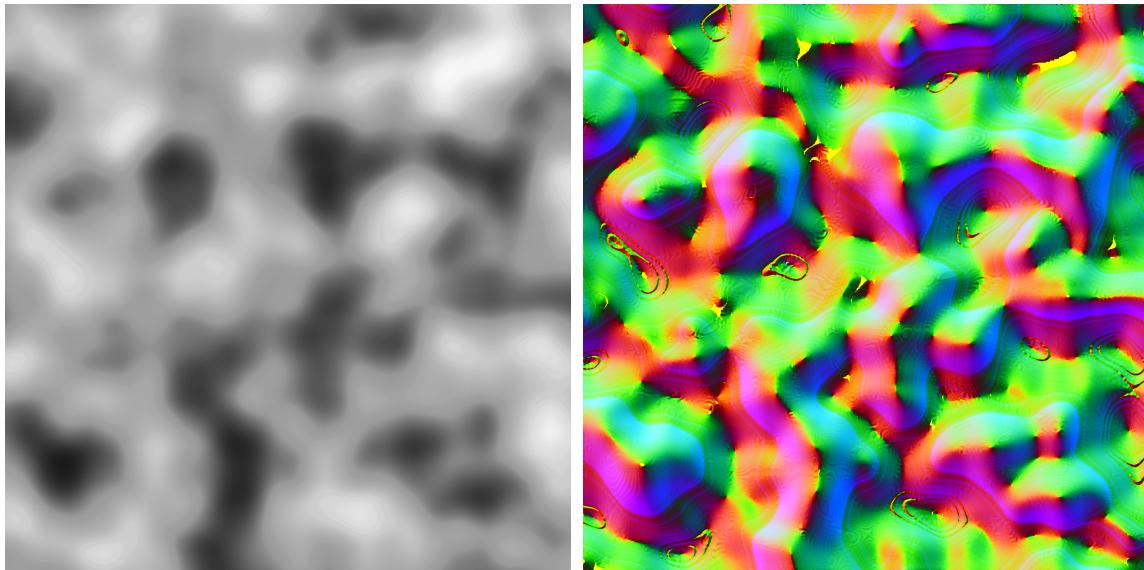


FIGURE 4 – Carte de gradients (à droite) de la carte d'altitude (à gauche)



### 3 Carte de rivières (algorithme descendant)

La carte de rivière est une image binaire qui indique pour chaque pixel s'il correspond à un rivière (blanc) ou non (noir).

L'idée est d'utiliser la carte de gradients pour faire "couler" de l'eau sur la carte, et ainsi "dessiner" les rivières.

Un premier algorithme que nous avons essayé de développer et d'implémenter utilise un "écoulement" descendant : on part des points de contacts mer/terre, et on remonte la pente jusqu'à arriver à une zone plate.

L'avantage d'un tel algorithme est qu'il suffit de le rendre récursif pour obtenir une arborescence de rivières (affluents).

Résumé simplifié de l'algorithme :

- On sélectionne un ensemble de points de départs : on utilise la carte binaire mer/terre pour détecter des points de contacts
- Pour chaque points de départ, on lance une méthode récursive : on va remonter la pente du terrain en lisant la carte de gradients et "dessiner" la rivière en dessinant des disques de diamètre correspondant à la largeur de la rivière
- Après une longueur minimale et si on atteint une zone suffisamment haute ou suffisamment plate, on va diviser la rivière en deux petites rivière, en relançant récursivement la méthode avec pour point de départ la position courante de la rivière additionné à un offset (pour éviter que les deux sous rivières ne restent "collées")
- Si la rivière a atteint la "largeur minimale" et qu'on atteint une zone suffisamment haute ou suffisamment plate, on arrête l'algorithme.
- Une fois que toutes les rivières ont été générées, on sélectionne les 'n' rivières les plus longues et on ignore les autres
- Lors de la génération de la carte de reliefs, on ajoute un nouveau type de relief qui correspond aux rivières.

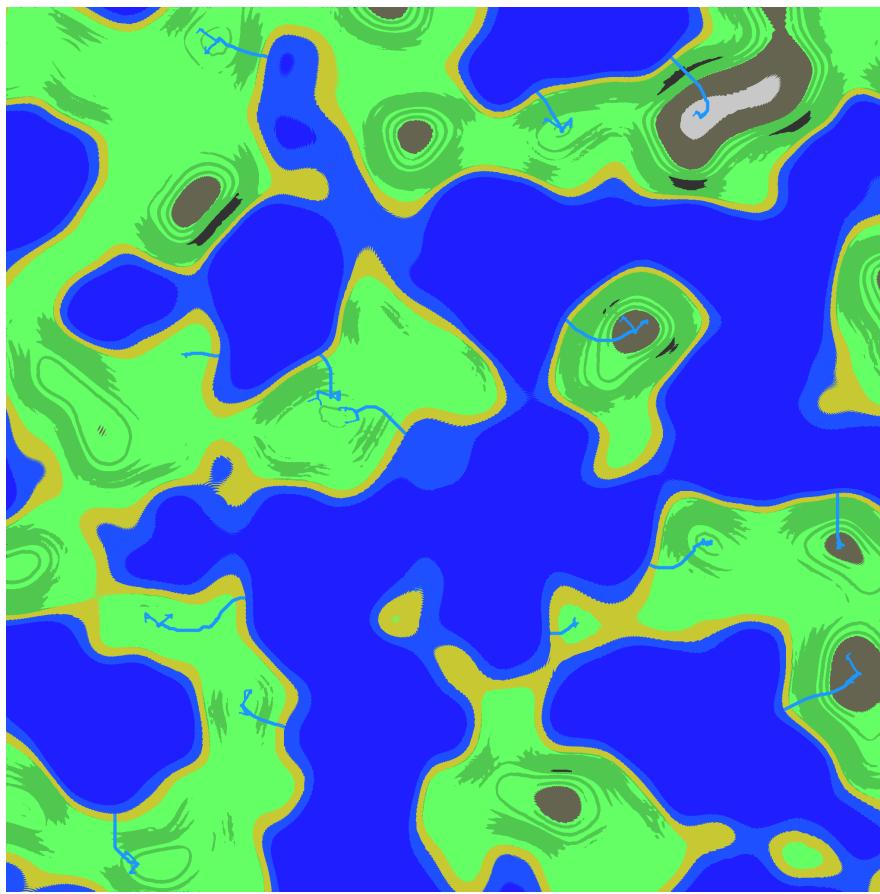


FIGURE 5 – Carte de rivière avec algorithme ascendant

Problème de cet algorithme :

- Il est difficile d'obtenir des résultats réalistes : comme on peut le voir sur la carte, certaines rivières s'arrêtent rapidement, ou sans atteindre de points élevés qui pourraient correspondre à des sources : on se retrouve avec des "bouts de rivières" qui souvent parcourent une plaine puis s'arrêtent.
- De la même manière, lorsque les rivières se divisent, les deux sous rivières suivent souvent des trajectoires étranges, notamment à cause de l'offset nécessaire.

La trajectoire d'une rivière est très chaotique, et il est donc probablement impossible de la reconstituer à l'envers.

Nous avons donc choisi d'essayer un autre algorithme, descendant cette fois ci :



## 4 Carte de rivières (algorithme descendant)

Cette version de l'algorithme est donc l'inverse du précédent : on part de points à haute altitude qui correspondent à des sources, et on fait "couler" l'eau en suivant la pente, jusqu'à atteindre la mer.

L'avantage d'un tel algorithme est qu'il est "physiquement réaliste" : on devrait obtenir des résultats plus satisfaisants. Aussi, il devrait facilement pouvoir être étendu à la génération de lacs.

Résumé simplifié de l'algorithme :

- On sélectionne un ensemble de points de départs : on utilise la carte d'altitude pour sélectionner obtenir une liste ordonnée de points, du plus haut au plus bas, puis on regroupe les points proches afin d'obtenir un ensembles de points assez éloignés les uns des autres
- Pour chaque point de départ, on lance 8 fois la méthode de dessin de rivière : on ajoute un offset au point de départ, afin d'obtenir 8 rivières qui partent de la source dans des directions différentes
- Comme précédemment, on suit la pente en lisant la carte de gradients, jusqu'à atteindre la mer ; si la rivière s'arrête avant de toucher la mer, on la supprime
- A chaque étape, on dessine un rond à la position actuelle de la rivière, de rayon correspondant à la largeur de la rivière ; dans le canal R, on inscrit 255, dans le canal G, on inscrit la largeur de la rivière, dans le canal B, on inscrit l'altitude du terrain (ces informations seront utiles plus tard)
- Pour simuler la combinaison de rivières affluentes, lorsqu'une rivière "touche" une autre rivière, on augmente sa largeur, afin de dessiner par dessus la rivière déjà présente et de l'élargir

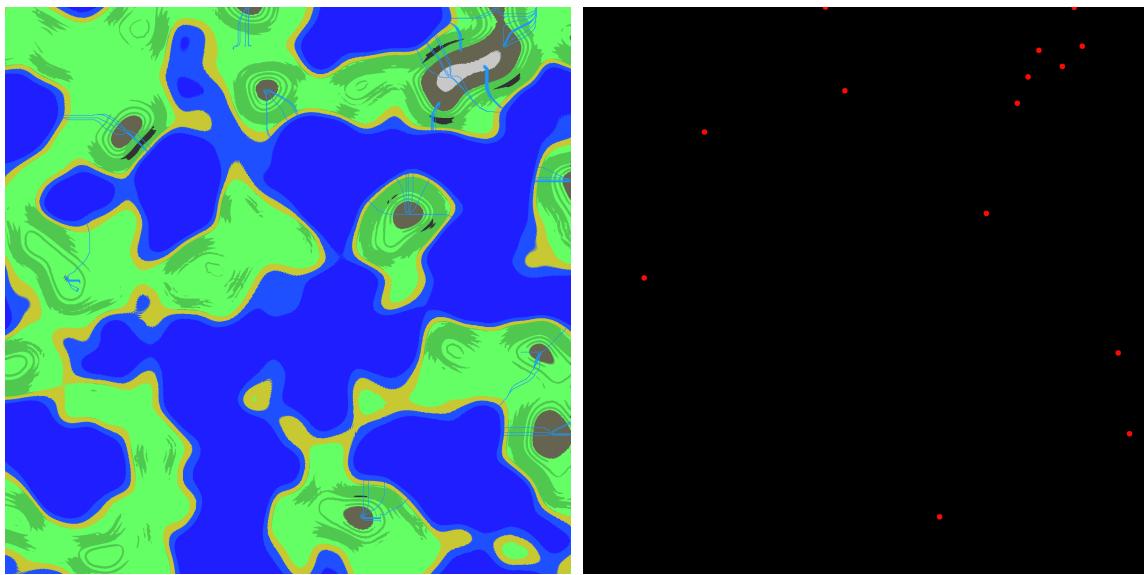


FIGURE 6 – Sélections de points de sources (à droite) et la carte de reliefs (à gauche). On peut voir que les sources se trouvent bien au niveau des points élevés (montagnes), et que les rivières coulent jusqu'à la mer

On peut voir que les sources sélectionnées ne correspondent pas aux rivières précédemment générées avec l'algorithme ascendant : on va donc obtenir un résultat différent (et probablement plus réaliste).

Pour ajouter des lacs, on va appliquer une suite de dilations à la carte de rivières. Pour chaque pixel non-nul (correspondant à une rivière), on va inscrire ses valeurs dans ses voisins, si la pente à la position du voisin est suffisamment faible.

Le seuil est calculé à partir d'une constante et est proportionnel à la largeur de la rivière (inscrite dans le canal G).

La valeur de largeur de la rivière inscrite dans les voisins est légèrement inférieure à la valeur de base (utilisation d'un coefficient légèrement inférieur à 1, par exemple 0.98), ce qui va diminuer le seuil lors de la prochaine itération, et donc empêcher les lacs de trop s'étendre.

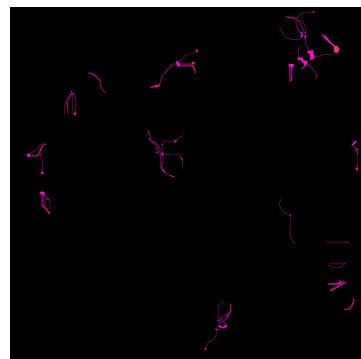


FIGURE 7 – Carte de rivières après création de lacs

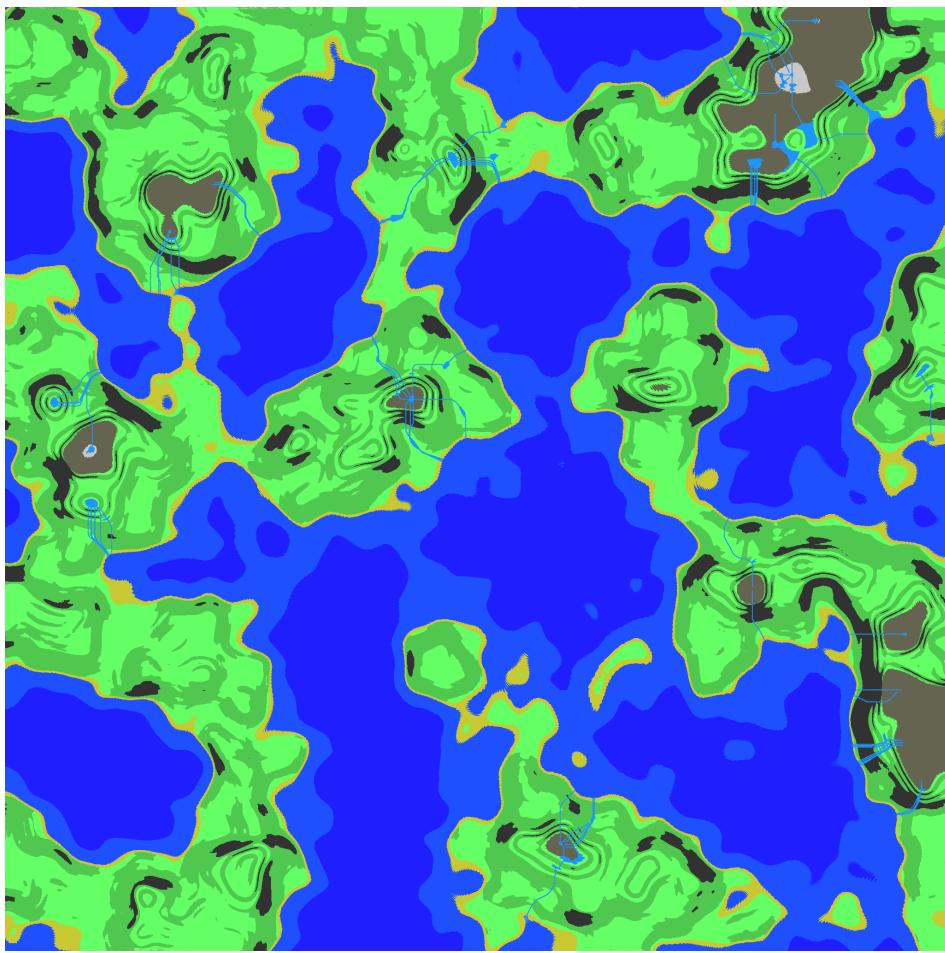


FIGURE 8 – Carte de reliefs après création de lacs (contexte différent des cartes précédentes)

## Améliorations possibles

Pour améliorer la trajectoire des rivières, et notamment favoriser leur regroupement (affluents), on pourrait indiquer pour chaque pixel (de la carte de rivière) la direction de la rivière la plus proche : ainsi, on pourrait "dévier" la trajectoire des rivières de sorte à ce qu'elles se rapprochent les unes des autres, et ainsi qu'elles fusionnent plus souvent.



## 5 Objectifs pour la semaine prochaine

### Détection de points d'intérêt

La prochaine étape est de parvenir à détecter des "points d'intérêt", c'est à dire des zones "intéressantes" sur la carte.

Cela peut être, par exemple, un lac, une cascade, une montagne, une île, une falaise, ect... Mais aussi, une zone particulièrement plate, sur laquelle on pourrait placer in-fine une ville.

Nous allons implémenter une première version, qui évoluera par la suite.

### Amélioration de la carte de reliefs

La carte de reliefs est pour le moment très simple, et ne recense que peu de reliefs différents.

L'objectif serait donc d'ajouter de nouveaux types de reliefs, et aussi générer une carte de relief indéxée (PGM) pour pouvoir associer un index à chaque relief.

### Interface graphique

Nous souhaiterions utiliser QT pour ajouter une interface graphique à notre application.