Distribution géométrique de régions de biomes

De la topologie vers la géométrie

Introduction

Au travers ce rapport, je vais présenter les quelques méthodes que j'ai pu étudier pour générer une géométrie à des régions de biomes à partir d'un graphe d'adjacence. Contrairement au précédent rapport (« Distribution géométrique de régions de biomes : de la géométrie à la topologie »), nous souhaitons voir un processus de génération de géométrie beaucoup plus déterministe, s'accordant avec le choix exact que l'utilisateur cherche à obtenir.

Entrées utilisateur

L'utilisateur fournit au programme un graphe topologique qui représente les adjacences qui devront se retrouver entre les régions géométriques finales. Les nœuds du graphe représentent une zone, une région à spatialiser. Si ce nœud est connecté à un autre nœud, on considère que les deux régions instanciées doivent se toucher en au moins un point (posséder une frontière commune).

Une condition est imposée pour que cela puisse se réaliser : on doit donner en entrée un graphe planaire. La définition même d'un graphe planaire est de pouvoir se représenter dans un plan en deux dimensions sans qu'aucune arête ne se croise. Plusieurs critères et algorithmes ont été établis pour déterminer la planarité d'un graphe (voir « Planarity Testing ») mais nous considérerons que l'utilisateur nous en fournit un.

L'utilisateur fournit également l'aire désirée pour chacune des régions.

Dans la pratique, nous insérons aléatoirement des points dans l'espace puis nous en extrayons le graphe d'adjacence calculé au travers d'une Triangulation de Delaunay. En effet, la question de la génération aléatoire de graphe planaire est encore très complexe. Cela nous permet en plus d'obtenir des valeurs pour les aires qui doivent être reconstructibles.

Remarque : Afin de palier au problème de région vide (ie. Une frontière entre plusieurs régions qui ne doivent pas se toucher), nous utilisons un graphe planaire triangulé. Cela signifie que tous les cycles dans le graphe ont une longueur d'exactement 3. Pour cela, un nœud « neutre » supplémentaire dans les cycles de longueur supérieure à 3 (« trous ») et raccordé à chaque nœud du cycle. Cela crée un « chordal graph », et donc sans région interdite.

Sortie

En sortie du programme, nous devrions retrouver un plan sur lequel sont réparties des formes géométriques représentant les nœuds du graphe d'entrée. Ces formes doivent respecter les contraintes d'aires et d'adjacences avec les régions voisines définies par le graphe d'entrée.

Méthodes

Plusieurs méthodes ont été essayées afin de résoudre ce problème. Aucune n'est totalement satisfaisante mais je vais tout de même les résumer ici. Les algorithmes sont divisés en 2 parties distinctes : tout d'abord le positionnement des graines initiales dans le plan pour représenter les nœuds, puis la création de la forme en elle-même autour de la graine pour générer la géométrie finale. Une 3ème partie s'insère entre les deux précédentes, qui est une optimisation de placement des graines.

Initialisation des graines

Méthode principale

Dans cette partie des algorithmes, nous souhaitons positionner les graines (la position initiale de la géométrie des régions) dans le plan à partir des informations que l'on possède dans le graphe. Chaque nœud possède un degré $deg(i)$ (nombre d'arêtes entrantes/sortantes) et chaque arête possède un poids qui représente ici une distance minimale entre deux nœuds. En effet, afin de récupérer une géométrie grossière, on peut considérer que les régions seront des sphères de rayon R (dépendant directement de l'aire désirée) ; donc le poids d'une arête est défini par $W_{ij} = R_i + R_j$.

La manière la plus répandue pour le dessin de graphe est d'insérer les nœuds dans le plan les uns après les autres en suivant un ordre appelé « Canonical Ordering » qui labellise les nœuds du plus « externe » vers le plus « interne ». Néanmoins nos méthodes fonctionnent principalement sur les essais-échecs pour fonctionner donc nous ne pouvons pas utiliser cet ordre invariant.

Dans notre méthode nous admettons une valeur d'entropie $e$définie par le degré et par le nombre de nœuds voisins déjà instanciés $\widetilde{deg}(i)$. La formule du calcul de l'entropie a été trouvée à tâtons et ne représente pas une formule mathématiquement correcte. Les meilleurs résultats sont trouvés avec

$e_i = \frac{1}{\widetilde{deg}(i)+\varepsilon}$ avec $\varepsilon > 0$ une très petite valeur pour éliminer la division par zéro.

D’autres formules ont été essayées sans avoir de résultats plus intéressants :

$e_i = \widetilde{deg}(i), e_i = \frac{deg(i)}{\widetilde{deg}(i)}, e_i = deg(i), \ldots$

Pour aller plus loin on pourrait calculer la distance avec tous les nœuds instanciés au lieu de se contenter de faire dépendre l’entropie aux voisins directs.

Nous choisissons maintenant l’un des nœuds avec la plus petite entropie et le positionnons dans le plan d’une manière proche de de la méthode de Poisson Disc Sampling. Si au moins un de ses nœuds voisins est déjà instancié, on le positionne à une distance $a.W_{ij} \leq d_{ij} \leq b.W_{ij}$des autres nœuds, avec $a, b \in \mathbb{R}\ a \leq 1 \leq b$une marge d’erreur souple.

Repositionner le point si les segments entre le nouveau point et ses voisins créent des intersections avec les arêtes déjà instanciées. Si aucune position ne satisfait la condition de non-croisement, replacer le germe au à la position créant le moins d’intersections.

Recalculer l’entropie des nœuds et recommencer le positionnement de chaque nœud.

L’étape d’optimisation des graines permettra parfois de corriger les erreurs de croisements.

Méthode alternative

Un algorithme différent a été implémenté : « Spectral graph drawing » basé sur les vecteurs propres de la matrice Laplacienne du graphe, mais les résultats sont très loin de ce que le papier associé peut proposer :

« Koren, Y. (2005). Drawing graphs by eigenvectors: Theory and practice. Computers and Mathematics with Applications, 49(11–12), 1867–1888. https://doi.org/10.1016/j.camwa.2004.08.015 »

Optimisation des graines

L’étape d’initialisation permet de positionner les germes dans le plan de manière à satisfaire les contraintes d’adjacences et d’aire en acceptant une certaine tolérance. L’étape d’optimisation raffine le placement des germes jusqu’à atteindre un minimum local.

On considère chaque nœud comme une particule libre attirée ou repoussée par les nœuds voisins du graphe d’adjacence associé. La recherche du point de stabilité est atteinte quand l’énergie$E = 0$en considérant l’énergie pour chaque particule étant la différence entre la distance souhaitée avec chacun de ses voisins et la distance effective avec ses voisins : $E_i = \sum_{j \in N(i)} d(i,j) - \hat{d}(i,j)$

Afin de converger plus rapidement vers cet équilibre, on utilise la distance L_2 :

$$E_i = \sum_{j \in N(i)} \quad E_{i,j} = \sum_{j \in N(i)} (d(i,j) - \hat{d}(i,j))^2 * sign((d(i,j) - \hat{d}(i,j)))$$

Au travers cette énergie, on réalise une descente de gradient en déplaçant la germe dans la direction normalisée $\overrightarrow{v_i} = \sum_{j \in N(i)} (P(i) - P(j)) * E_{i,j}$

Ce déplacement itératif des particules converge rapidement vers un minimum local.

Il est possible que cette optimisation de longueur des arêtes rentre en conflit avec la contrainte de non-croisement. Deux solutions peuvent être appliquées pour contrer ce problème :

1. Recommencer l'initialisation jusqu'à obtenir une solution stable (ce qui a été fait dans ce travail)
2. Considérer les arêtes formées par les autres nœuds comme des obstacles qui pénalisent l'énergie lorsque l'on s'y dirige. La quantité de calculs est possiblement plus importante que de recommencer tout le processus, et le point de stabilité ne peut jamais atteindre $E = 0$. De ce fait, je n'ai pas investigué plus loin dans cette direction.

Génération de la géométrie

Dans les étapes précédentes nous avons pu positionner les germes (les nœuds du graphe) dans le plan en respectant au mieux les contraintes de non-croisement des arêtes, ce qui nous garantit qu'une géométrie des régions sans superpositions entre elles mais avec des adjacences est possible.

Dans cette partie de la méthode nous allons générer des formes géométriques à partir des germes précédemment instanciés qui doivent modeler des surfaces dont l'aire est définie par l'utilisateur dans le graphe initial, qui doivent partager une frontière avec les régions adjacentes du graphe mais aucune frontière avec les nœuds non-adjacents.

La méthode applique principalement la technique de Region Growing en initialisant chaque région comme étant une forme circulaire de rayon négligeable composée d'un nombre fixe de sommets. Cette forme est alors agrandie en déplaçant chaque somment le long de sa normale jusqu'à collision avec une autre région ou jusqu'à ce que la région ait l'aire désirée.

Néanmoins plusieurs stratégies différentes ont été essayées pour arriver plus efficacement à un résultat où toutes les contraintes d'adjacences sont respectées. Pour cela on peut essayer de faire varier la normale que chaque sommet doit suivre, la vitesse de déplacement de chaque sommet ou encore le comportement des régions lors de collisions.

Stratégie naïve

Une première stratégie très « naïve » est d'agrandir chaque forme dans le sens des normales de manière uniforme en s'arrêtant lors de collision. On comprend rapidement que les régions centrales n'ont pas le temps d'atteindre leur aire optimale contrairement aux régions extérieures et le résultat n'est ni plus ni moins qu'un diagramme de Voronoï discrétisé.

Stratégie du plus petit l'emporte

La seconde stratégie du « plus petit l'emporte » autorise une région à en écraser une autre dans la condition où elle est en retard dans son développement. La région $R_i$entre en collision avec la région R_j et on calcule le ratio d'aire effective par rapport à l'aire désirée $a(R_i) = \frac{A(R_i)}{\hat{A}(R_i)}$. Si $a(R_i) < a(R_j)$ alors les sommets en frontière entre $R_i$et R_j sont déplacés dans la direction de la normale de R_i le long du vecteur $\vec{v} = \overrightarrow{v_\iota}\frac{a(R_i)}{a(R_j)}$. Par cette stratégie toutes les régions voient leur aire converger vers l'aire désirée, néanmoins la contrainte d'adjacence n'est ni garantie, ni optimisée.

Stratégie d'augmentation variable

La stratégie « d'augmentation variable » a pour but d'élargir la forme de la région dans l'objectif de remplir au plus vite les contraintes d'adjacences. Pour chaque sommet de la région, étudier si l'extension de la région dans la direction de sa normale présente un intérêt. Pour cela on compare le produit scalaire entre la normale $\vec{n}$ et les sommets de chaque région adjacente $\overrightarrow{v_j}$ pour obtenir le score d'importance $I_i = \sum_{R_j \in N(R_i)} \sum_{\vec{v}_j \in V(R_j)} \vec{n} \cdot \overrightarrow{v_j}$

On peut ensuite déplacer chaque sommet de la région par le vecteur de déplacement pondéré par l'importance de tous les autres sommets $\overrightarrow{v_i} = \vec{v} * \frac{I_i}{\sum_{j \in V(R_i)} I_j}$.

On a alors des régions qui optimisent les relations d'adjacences tout en laissant leur aire converger vers la valeur souhaitée. La stratégie souffre tout de même d'une incapacité à faire s'adjoindre deux régions si elles sont divisées par une troisième région.

Une amélioration possible serait de laisser un sommet transmettre son importance à ses voisins directs si la demi-droite passant par le sommet et dirigée par la normale du sommet intersecte une région qui ne doit pas être adjacente ou dont la condition d'adjacence est déjà satisfaite.

Stratégie d'augmentation dirigée

La stratégie « d'augmentation dirigée » fonctionne de manière similaire à la précédente « augmentation variable ». Les sommets d'une région voient la direction de leur normale déviée par rapport à la normale originale dans le but de s'étendre plus efficacement voire de contourner des régions obstruant l'adjacence de deux nœuds du graphe.

La nouvelle normale $\vec{\vec{n}}$ est orientée vers les sommets les plus proches de chaque région adjacente en fonction de l'orientation de la normale initiale $\vec{n}$ et de la distance qui les séparent.

Le calcul proposé est $\vec{\vec{n}} = \left\| \sum_{R_j \in N(R_i)} \frac{\vec{v}_j}{\|\vec{v}_j\|} * \left(\vec{n} \cdot \vec{v}_j\right)^{\alpha} * \|\vec{v}_j\|^{\beta} \right\|$ avec $\vec{v}_j$ le vecteur entre le sommet courant et le sommet le plus proche de la région adjacente R_j et α, β des exposants dont le but est d'ajuster l'importance de l'orientation initiale et/ou la distance entre les régions.

Bien que cette stratégie aide beaucoup à l'optimisation des contraintes d'adjacences sans entraver l'expansion de l'aire, surtout lorsqu'elle est couplée à la stratégie précédente, les formes gardent tout de même des difficultés pour contourner de gros obstacles.

Stratégie d'augmentation par recherche de chemin le plus court

Il semble plutôt logique au vu des lacunes que présentent les dernières stratégies proposées qu'une recherche de chemin le plus court puisse être envisagée pour contourner une région qui s'interpose entre deux régions adjacentes.

Autour de chaque région une grille de taille $w \times w$ est définie suffisamment grande pour englober les régions adjacentes, avec des obstacles à chaque coordonnée correspondant à l'une des autres régions à contourner. Un algorithme de recherche de chemin le plus court est lancé pour chaque région pour chaque région adjacente à chaque pas de temps.

Pour chaque sommet à déplacer, on observe le chemin le plus court le reliant à une région adjacente, et on le déplace dans la direction du chemin.

Cette stratégie me semble logiquement être la plus efficace, mais la lenteur d'exécution demande une bonne optimisation de la résolution de la grille ($w \times w$) pour être à la fois suffisamment précis et rapide. On a en effet une complexité $O(|E|)$ pour l'algorithme de A*, soit $O(w^2)$ dans notre cas. Répêté pour chaque région pour chaque région adjacente, on obtient une complexité $O(w^2 * |R|^2)$ avec $|R|$ le nombre de régions.

Dans la précipitation, j'ai même lancé cet algorithme pour chaque sommet des régions, soit une complexité $O(w^2 * |R|^2 * |S|)$. Avec une meilleure implémentation et en utilisant un algorithme de Floyd-Warshall en temps $\Theta(|V|^3) = \Theta(w^6)$ ou Johnson en temps $O(|V|^2 \log|V| + |V||E|) = O(w^4 \log w + w^4)$, il est peut-être possible d'obtenir des temps de calcul réalistes.

Résultats

En cours.

Améliorations possibles