Cahier des charges du projet M2-GA : maillage de Delaunay d'un terrain

J-M. Moreau

31 octobre 2014

Objectif général

On souhaite modéliser un terrain 2D$\frac{1}{2}$ à partir de données altimétriques et en proposer une visualisation tri-dimensionnelle en utilisant un maillage triangulaire sur le principe de Delaunay. La demande minimum est que les points soient générés avec un loi uniforme pour les abscisses, les ordonnées et les altitudes, que le rendu soit en faces pleines et qu'il soit possible de se déplacer en trois dimensions à l'aide de la souris autour du maillage, sous OpenGL et en code source C (pas de C++). L'affinage du maillage consiste en l'insertion de points choisis en fonction de leur éloignement vertical au terrain (file de priorité) jusqu'au premier des trois critères suivants, selon les options sur la ligne de commande : tous les points sont insérés, ou bien le nombre de facettes est supérieur à une valeur décidée par l'utilisateur, ou enfin l'adéquation du maillage au terrain est passée en-dessous d'un seuil déterminé en entrée, qui correspond à un pourcentage de l'amplitude maximum des altitudes du terrain.

Génération aléatoire des points et de leurs altitudes

On part du carré unité $]0,1[^2$, noté $ABCD$ (ordre trigonométrique de succession sur la frontière), en 2-D 1/2, triangulé par une de ses deux diagonales, disons AC.

Obligatoire : On génère $n-4$ points (ou "sites candidats") aléatoirement selon une loi uniforme dans $]0,1[^2$, dans l'intérieur strict du carré. Il y a donc en tout n points générés. Les sommets du carré, ainsi que tous les sites candidats possèdent une altitude "aléatoire", générée selon une distribution par défaut uniforme dans un intervalle fixé par des constantes absolues. On pourra fixer le nombre de sites candidats au lancement (donc le nombre n de points à générer en tout), sur la ligne de commande. Exemple : `delaunay -n5000`.

Optionnel : on pourra de plus utiliser d'autres lois de distribution comme support du phénomène de génération en x, y d'une part et en z (altitude) de l'autre. Par exemple, les altitudes peuvent être déduites d'une surface 2D$\frac{1}{2}$ modèle, d'une carte de hauteurs à partir d'un bitmap ou d'un vrai terrain. Les abscisses et ordonnées peuvent etre générées selon des lois gaussiennes, avec ou sans *clusters* locaux. On peut encore imaginer la génération de terrains fractals, avec perturbation locale de l'altitude. Dans tous les cas, la coloration neige en fonction de l'altitude est du plus bel effet (optionnellement optionnel, évidemment)...

Chaque possibilité s'accompagnera d'une option à la ligne de commande, avec ses éventuels paramètres propres (par exemple, moyenne et écart type de la loi de distribution).

Note : Cette partie du travail est plutôt réservée aux trinômes, exclue pour les monômes et à n'envisager, pour les binômes, qu'une fois tout le reste fait.

Affinage et construction de Delaunay

Au départ, chacun des deux triangles initiaux « abrite » tous les points p du nuage tels que la projection sur le plan $z = 0$ de p est contenue dans la projection du triangle (dans le même

plan $z = 0$). Cette propriété d'« appartenance » sera vraie pour tout triangle du maillage à tout instant. Ainsi, il est naturel de définir, pour tout triangle Δ et tout point p qui lui appartient, la distance verticale $d_\Delta(p)$ comme la différence entre l'altitude de p et l'altitude de son projeté sur le plan Π_Δ défini par les trois sommets du triangle.

A tout moment, il doit être possible de savoir en temps constant, pour tout triangle Δ déjà construit dans le maillage, quel point candidat p_Δ qu'il contient au sens précédent réalise la distance verticale $d_\Delta(\cdot)$ la plus grande. Il est inutile d'organiser les sites candidats d'un même triangle en autre chose qu'une simple liste chaînée dont le premier élément est l'extremum pour la distance verticale. Cette liste, qui permet insertion en tête et en queue en temps constant, mais ne nécessite pas de suppressions individuelles, est constituée une fois pour toutes au moment de la création de tout triangle et n'est pas remise en cause tant que ce dernier reste dans le maillage. Si le triangle disparaît, il est supprimé du maillage, remplacé par (l'union de) trois autres, et sa liste de points candidats est simplement redistribuée parmi les trois nouveaux triangles.

En organisant les triangles déjà constitués en file de priorité fondée sur la distance verticale [1], à chaque itération du programme on extrait de la file le triangle qui renferme le point candidat de plus grand écart vertical parmi tous (donc le triangle qui est en première position de la file de priorité, et donc dont le point candidat en tête de liste chaînée est plus éloigné verticalement de sa portion de terrain que tout autre point candidat ne l'est de la sienne, qu'il appartienne à ce triangle ou non), on insère son point candidat extremum comme site dans la triangulation déjà obtenue, on supprime le triangle dans la triangulation et on le remplace par trois triangles partageant ce site comme sommet, et les arêtes non détruites du triangle supprimé, puis on s'assure que la nouvelle triangulation est de Delaunay (technique incrémentale itérative à base de pile, *cf.* cours).

Attention : la difficulté majeure provient du fait que l'on est amené à effectuer des échanges de diagonales (*diagonal flips*) dans les quadrilatères convexes formés par deux triangles adjacents par une diagonale du quadrilatère. Chaque échange correspond à la suppression de deux triangles et leur remplacement dans la triangulation par deux triangles adjacents par la seconde diagonale de ce quadrilatère. Il n'est pas raisonnable de supprimer deux triangles dans une file de priorité implémentée par tableau (essayez de voir les difficultés), mais bien plus simple de garder chacun dans la file, en modifiant ses propriétés (nouveaux sommets, nouvelle liste de sites candidats à partir des deux anciennes, et nouveau site candidat extremum). La modification de la valeur de distance verticale maximale provoque soit l'enfoncement (*downheap*) du triangle modifié, soit son élévation (*upheap*), soit encore son maintien là où il est, de manière mutuellement exclusive. Il est primordial que cette opération soit réalisée en temps logarithmique !

Critère d'arrêt

On procède ainsi jusqu'à ce que :

soit tous les points candidats générés aléatoirement soient insérés (absence ou non satisfaction des deux critères ci-après)

soit la modélisation atteigne un nombre maximum de facettes triangulaires fixé à l'avance sur la ligne de commande. Exemple d'appel : `delaunay -f500`.

soit l'écart maximal de tout point au plan du triangle qui le contient soit inférieur à un seuil fixé (en fonction de l'échelle ?), déterminé sur la ligne de commande. Exemple d'appel : `delaunay -s0.01`. Attention : ce critère doit être accompagné d'une logique bien exprimée. Par exemple le seuil peut être une fraction ou un pourcentage de l'amplitude verticale maximale. Lui donner une valeur de 0.01 ne veut rien dire en soi : explicitez votre choix.

Ces critères d'arrêt ne sont pas mutuellement exclusifs. Ils sont tous obligatoires pour le projet. D'autres, optionnels, seront envisageables (densité locale variable, etc).

1. La taille maximale de cette file de priorité est linéaire et connue à l'avance, selon les formules vues en cours.

Affichage

Par défaut le programme utilisera les données et affichera, sans qu'il soit besoin de demander quoi que ce soit de particulier, le maillage obtenu en vue 3D, faces pleines (colorées, texturées ou non), avec possibilité de zoom rentrant/sortant, et de rotation autour de la surface reproduite, à l'aide de la souris et de sa roulette voire avec l'aide du clavier. De plus, il est demandé de proposer un mode sans affichage graphique (option explicite sur la ligne de commande). S'il est nécessaire de développer, pour des soucis de debogage, un mode d'insertion en pas à pas, il devra être lancé par une option explicite de la ligne de commande, le mode par défaut étant l'insertion complète et ininterrompue des points jusqu'à satisfaction du ou des critères d'arrêt listés plus haut. Enfin, il est demandé de pouvoir, à l'aide d'une option spécifique sur la ligne de commande, visualiser le maillage en 2D (vue de dessus) et en fil de fer. L'idée de pouvoir basculer d'un mode d'affichage ($2D\frac{1}{2}$ faces pleines) à l'autre (2D fil de fer) par appui sur une touche spécifique (avec retour à la vue initiale après deux pressions) est tout-à-fait bienvenue.

Outre l'affichage graphique, il est demandé d'afficher sur la console le nombre de point candidats insérés, le nombre de facettes générées, le temps global d'exécution hors graphique, la taille maximum de la pile pour Delaunay atteinte pendant l'exécution et enfin le rapport à $\log m$ (base 2) du nombre de comparaisons entre triangles dans les opérations primitives (down ou up heap) sur la file de priorité pendant la totalité de l'exécution, m étant le nombre de points candidats effectivement insérés (valant n si on ne limite pas le nombre de facettes).

Pour les trinômes

Outre les générations aléatoires exotiques et optionnelles mentionnées plus haut, il est demandé aux trinômes de donner la possibilité de localiser (cf. dernier CM), en mode d'affichage 2D (vue de dessus, fil de fer), tout point indiqué à la souris en affichant le triangle hôte avec une (absence de) couleur ad hoc, puis de laisser l'utilisateur recommencer, jusqu'à appui sur une touche spécifique marquant la fin du « jeu ». Pour les très forts, même question, mais en 3D !

Pour les monômes

Génération de points, construction puis affinage du maillage, respect des conditions d'arrêt, affichage 2D sans zoom ni déplacements.

Organisation générale

Langage C, OpenGL. La documentation interne sera effectuée à l'aide de `doxygen`. Le rapport (une dizaine de pages explicatives sur les méthodes et leurs complexités théorique et pratique) contiendra une présentation du travail, la part de chacun, et une explication sur les choix effectués pour les structures et algorithmes principaux. Pas de listing. Le rapport sera rédigé en LaTeX.

Note importante : que vous travailliez sur une machine personnelle ou non, le code source devra être compilable et le programme exécutable sur les machines graphiques de Nautibus en salle TP (condition non-négociable) ainsi que sur la mienne : partez du makefile fourni, qui garantit cela.

Vous rendrez une archive `tar-gz` du répertoire contenant votre projet (retirer tous les objets et exécutables avant). Le dossier portera comme nom les initiales des auteurs. Il contiendra explicitement un README qui permettra de comprendre comment exécuter le programme (options, données, etc), où trouver les données, etc (petit manuel de l'utilisateur).

Organisation de l'arborescence de travail

`xx-yy-zz`

```
SOURCES   DATA   DOCS  OBJS  makefile  README  RAPPORT delaunay

xx-yy-zz: initiales des auteurs.
SOURCES: .c, .h.
DATA: fichiers de données éventuelles.
OBJS: .o et exécutable
makefile: fichier de commandes maître, pilotant la compilation du programme,
   la compilation de la documentation, et celle du rapport. cf. make -C ...
README:: cf. plus haut.
RAPPORT: les fichiers LaTeX pour la génération du rapport en pdf.
delaunay: lien symbolique sur l'exécutable dans OBJS.
```

L'archive (`xx-yy-zz.tgz`) sera envoyée dans un mél à mon adresse

`Jean-Michel.Moreau@univ-lyon1.fr`,

de sujet (impérativement) :

Projet GA-M2 : ...

où ... symbolise les noms des auteurs du projet. Le corps du message listera les noms complets et les adresses mél où chacun des auteurs seront susceptibles d'être joints **même** pendant leur stage : c'est à cette époque-là que je commence à vraiment regarder les projets et vous ne lisez souvent plus vos méls à la fac !

Calendrier

Diffusion du sujet : 31 octobre 2014

travail binômes pendant TPs : à partir du 7 novembre 2014.

Rendu effectif du rapport et archive du projet : à déterminer par accord sur la promo. Généralement juste après les congés de Noël.

Soutenance par binômes : Généralement après congés de Noël et avant la reprise des cours.