

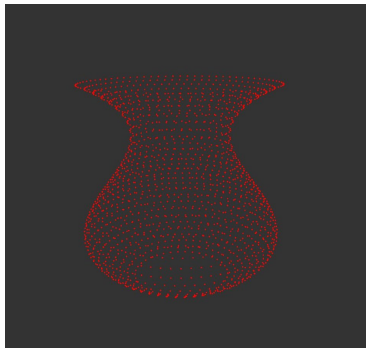
# Reconstruction & Géo Algo

Romain Vergne

UJF

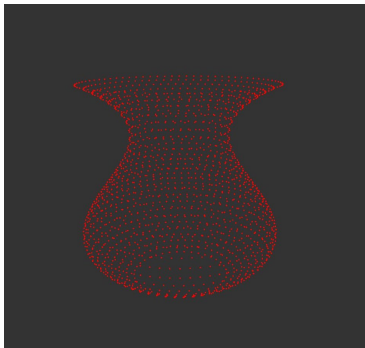
2014/2015

# RECONSTRUCTION DE DONNÉES SPARSE

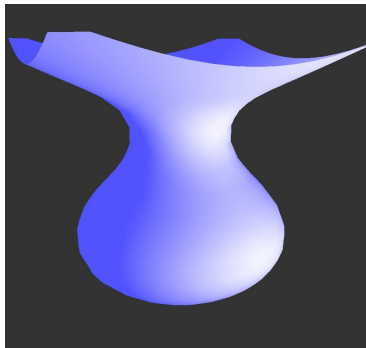


Entrée : Points

# RECONSTRUCTION DE DONNÉES SPARSE



Entrée : Points



Sortie : Surface

# LES FICHIERS

Vous disposez des classes suivantes (+ fonctions de base) :

- `geom` : classe `Point3D`
  - produits scalaire/vectoriel/distances/norme/...
  - liste de points (`v_Point3D`)
  - vecteurs propres d'une matrice 3x3 symétrique
- `data_struct_algo` : classes `Graphe`, classe `Grille3D`
  - création `Graphe`/calcul arbre couvrant minimal/...
  - initialisation d'une grille3D pour la fonction implicite
- `iso_value` : classe `SurfacelsovaleurGrille`
  - calcule le marching tetrahèdre à partir d'une grille + une fonction implicite
- `viewer / eventWidget` : fenêtre et menu

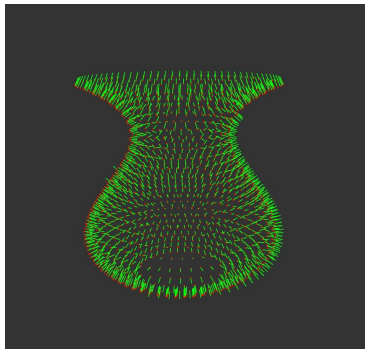
# LES FICHIERS

- `pointsToSurface` :
  - la **SEULE** classe à modifier !
  - une liste de points est donnée : `v_Point3D _points` ;
  - Vous devrez y remplir des structures/fonctions prédéfinies...
    - `v_Point3D _noNormals` ; // (computeNonOrientedNormals)
    - `v_Point3D _oNormals` ; // (computeOrientedNormals)
    - Graphe `_acm` ; // (computeMinimalSpanningTree)
    - implicit function (computeImplicitFunc)
  - ... et obtenir la surface finale :
    - `v_Triangle3D _surfacep` ; (computeMesh)
    - `v_Triangle3D _surfacen` ; (computeNormalsFromImplicitFunc)

# MÉTHODE IMPLICITE

- 1 Calculer des normales non orientées
- 2 Calculer l'arbre couvrant de poids minimum
- 3 Réorienter des normales
- 4 Déterminer une fonction  $f(x, y, z)$  en tout point
- 5 Calculer la surface isovaleur  $f(x, y, z) = 0$
- 6 Calculer les normales (finales) de la surface

# 1 : CALCUL DES NORMALES NON ORIENTÉES



# 1 : CALCUL DES NORMALES NON ORIENTÉES

REMPLIR `_noNormals` / FONCTION “`COMPUTENONORIENTEDNORMALS`”

- Pour chaque sommets  $p$ 
  - 1 déterminer les  $k$ -voisins  $pts$
  - 2 calculer le barycentre  $B$  des points  $pts$
  - 3 calculer la matrice  $A = pts - B$
  - 4 calculer la matrice  $A^T A$
  - 5 calculer les vecteurs propres associés à

$$A^T A = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{12} & A_{22} & A_{23} \\ A_{13} & A_{23} & A_{33} \end{pmatrix}$$

- 6 y récupérer les normales non orientées `_noNormals`



# 1 : CALCUL DES NORMALES NON ORIENTÉES

REMPLIR `_noNormals` / FONCTION “`COMPUTENONORIENTEDNORMALS`”

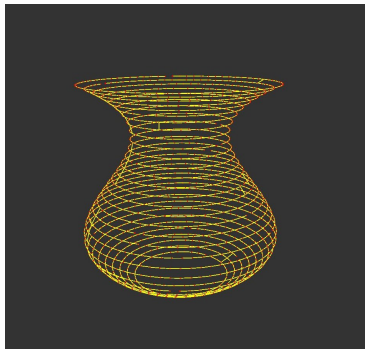
- Pour chaque sommets  $p$ 
  - ➊ déterminer les  $k$ -voisins  $pts$
  - ➋ calculer le barycentre  $B$  des points  $pts$
  - ➌ calculer la matrice  $A = pts - B$
  - ➍ calculer la matrice  $A^T A$
  - ➎ calculer les vecteurs propres associés à

$$A^T A = \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{12} & A_{22} & A_{23} \\ A_{13} & A_{23} & A_{33} \end{pmatrix}$$

- ➏ y récupérer les normales non orientées `_noNormals`

Astuce : utiliser les routines “`kneighborhoodPoints`” (`pointsToSurface`) et “`calcul_repere_vecteurs_propres`” (`geom`) permettant de récupérer le repère local à partir de la matrice  $A^T A$

## 2 : ARBRE COUVRANT DE POIDS MINIMUM

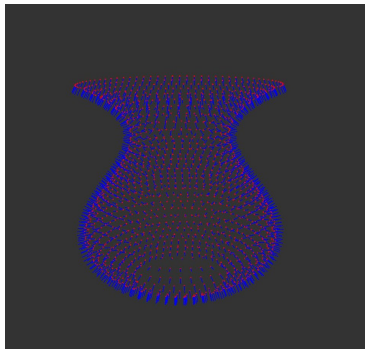


## 2 : ARBRE COUVRANT DE POIDS MINIMUM

REMPLIR `_acm` / FONCTION “`COMPUTEMINIMALSPANNINGTREE`”

- ❶ calculer le graphe de proximité  $G$  des points `_points`
  - un arc = paire de points  $p_i, p_j$ , avec
  - $distance(p_i, p_j) < r$ ,  $r$  un rayon
  - ATTENTION au rayon choisi :
  - le graphe doit avoir UNE SEULE composante connexe
- ❷ calculer le graphe couvrant de poids minimum à partir de  $G$  :  
`_acm`
  - utiliser “`arbre_couvrant_minimal`” (`data_struct_algo`)

### 3 : RÉORIENTATION DES NORMALES

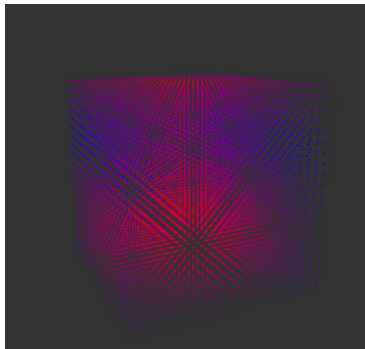


# 3 : RÉORIENTATION DES NORMALES

REMPLIR **\_ONORMALS** / FONCTION "COMPUTEORIENTEDNORMALS"

- ❶ pour le noeud racine de l'arbre courant minimum **\_acm**
  - ❶ regarder les normales entre le noeud père et les noeuds fils
  - ❷ si besoins, réorienter les normales des noeuds fils
  - ❸ recommencer le même procédé en partant des noeuds fils

## 4 : LA FONCTION IMPLICITE



## 4 : LA FONCTION IMPLICITE

REEMPLIR LA FONCTION "COMPUTEIMPLICITFUNC"

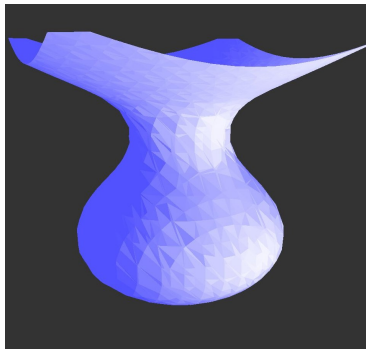
Pour un point donné  $\mathbf{x} = (x, y, z)$ , la fonction renvoie la valeurs de la fonction implicite en utilisant MLS. A partir de points  $\mathbf{x}_i$  + normales  $\mathbf{n}_i$ , calculer la distance moyenne aux plans :

$$f(\mathbf{x}) = \frac{\sum \mathbf{n}_i^T (\mathbf{x} - \mathbf{x}_i) w_i(\mathbf{x})}{\sum w_i(\mathbf{x})}$$

$$w_i(\mathbf{x}) = w(\|\mathbf{x} - \mathbf{x}_i\|), \text{ et } w(x) = e^{-\left(\frac{x}{\sigma}\right)^2}$$

ATTENTION au  $\sigma$  utilisé (dépend de la taille de l'objet)

## 5 : SURFACE ISOVALEUR



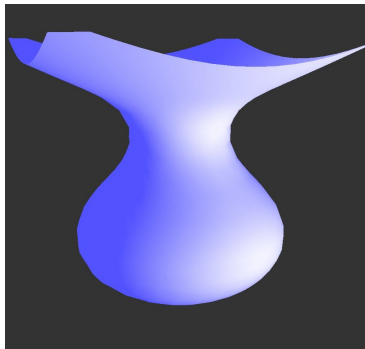


## 5 : SURFACE ISOVALEUR

REMPLIR **\_SURFACEP** / FONCTION “COMPUTEMESH”

- ❶ créer une Grille3D  $G$  que l'on utilisera pour le marching
  - déterminer la boîte englobante (doit contenir l'objet - et même un peu plus)
  - déterminer le pas de la grille dans les 3 dimensions
  - trop petit pas : calculs qui suivront trop lent
  - trop grand pas : mauvaise reconstruction
- ❷ créer un tableau  $v$  contenant les valeurs de la fonction implicite
  - même taille que la grille !
  - contient les valeurs de la fonction implicite pour chaque position de la grille3D (appeler “computeImplicitFunc” implémenté précédemment)
- ❸ calculer la surface isovaleur **\_surfacep**
  - utiliser SurfaceIsovaleurGrille - fonction “surface\_isovaleur” (iso\_value)

## 6 : NORMALES FINALES



## 6 : NORMALES FINALES

REMPLIR **\_SURFACEN** / FONCTION "COMPUTENORMALSFROMIMPLICITFUNC"

On pourrait prendre les normales estimées originales : discontinuités.  
Solution : calculer la dérivée de la fonction implicite en chaque sommet de la surface. La dérivée du champs scalaire donne la normale !  
Elle se calcule de la manière suivante :

- pour chaque point  $\mathbf{p}_i(x_i, y_i, z_i)$  de la surface faire
  - $n_x = f(x_i - 0.01, y_i, z_i) - f(x_i + 0.01, y_i, z_i)$
  - $n_y = f(x_i, y_i - 0.01, z_i) - f(x_i, y_i + 0.01, z_i)$
  - $n_z = f(x_i, y_i, z_i - 0.01) - f(x_i, y_i, z_i + 0.01)$
  - normaliser( $n_x, n_y, n_z$ ) et l'ajouter dans **\_surfacen**

# ALGORITHME

- 1 Calculer des normales non orientées
- 2 Calculer l'arbre couvrant de poids minimum
- 3 Réorienter des normales
- 4 Déterminer une fonction  $f(x, y, z)$  en tout point
- 5 Calculer la surface isovaleur  $f(x, y, z) = 0$
- 6 Calculer les normales (finales) de la surface

# ALGORITHME

- ❶ Essayer plusieurs valeurs de lissage
- ❷ Essayer plusieurs valeurs de pas pour la grille
- ❸ Essayer avec plusieurs fichiers de points ( voir data/ )

# A RENDRE

- un dossier compressé prenom-nom.tgz contenant :
  - 1 un rapport au format pdf (images + explications)
  - 2 le fichier pointsToSurface.h
  - 3 le fichier pointsToSurface.cpp

# CONSEILS

- n'hésitez pas à enrichir la classe PointsToSurface
  - avec des variables à vous
  - avec des fonctions temporaires
- utilisez les fonctions/variables à votre disposition
  - boundingBox donnée
  - distance minimale moyenne donnée
  - etc...

# COMPILATION

- bibliothèques :
  - QT
  - OpenGL
  - libqglviewer (à installer si nécessaire)
- Préparation :
  - éditer main.pro
  - changer les chemins / libs si nécessaire
- compiler :
  - qmake && make
- tester :
  - ./pointsToSurface data/file.txt



# BIBLIO

- Cours de Nicolas Szafran 2011/2012  
(<http://www-ljk.imag.fr/membres/Nicolas.Szafran/>)
- Scattered Data Interpolation and Approximation for Computer Graphics (Siggraph Asia course 2010)
- Implicit surface reconstruction from point clouds (thèse de Johan Huysmans)
- The Method of Least Squares (Steven J. Miller)
- Cours de Gael Guennebaud (moving least squares) :  
<http://www.labri.fr/perso/guenneba/>