Chapitre 4: Implémentation

I. Introduction

Le but de notre travail est de réaliser une application qui permet de faire la décomposition d'un objet 3D en régions, afin d'être utilisées dans les domaines qui interagissent avec la 3D, tel que le traitement des données tridimensionnelles.

II. Choix techniques

II.1 Le langage de programmation

Ce projet étant essentiellement centré autour de la réalisation d'une interface graphique, notre choix s'est naturellement porté vers des langages orientés objet : Java et C++. Il est donc nécessaire de faire un choix.

Notre projet en plus de l'interface graphique, nécessite une bibliothèque graphique qui permet de développer des applications 3D. Il existe plusieurs bibliothèques en C++ notamment OpenGl.

Mais l'existence de la bibliothèque graphique VTK en java en plus du fait que l'un des principaux avantages de Java est sa portabilité native, contrairement à C++, a finalement confirmé notre choix.

II.2 L'environnement de développement

Il existe une multitude d'environnements de développement (IDE) qui prennent en charge les langages Java et C++. Parmi eux, seul un nombre plus restreint permet une utilisation plus rapide et efficace, Eclipse et Visual Studio offrent tous deux des outils de qualité professionnelle.

Cependant, Eclipse a l'avantage d'offrir des fonctionnalités telles que la vérification de la syntaxe en cours de frappe, ainsi que le « « refactoring [10] ». et en plus de sa portabilité c'est un logiciel gratuit contrairement à Visual Studio, donc on a écarter ce dernier comme outil de développement.

III. Format des données d'entrée

Il existe un grand nombre de type de fichiers concernés par la 3D, nous avons eu l'occasion d'en décrire quelques uns dans le chapitre II, et après avoir fait un tours sur

[10] Le refactoring est une opération qui consiste en la maintenance du code d'un programme. Pour un langage de programmation orientée objet le refactoring peut par exemple permettre de renommer des membres de classe et de répercuter les changements dans tout le code source (commentaires inclus). Le refactoring est couramment employé pour nettoyer le code.

ces types, notre choix s'est porté sur les fichiers de type VTK a cause de la simplicité de la structure des points 3D et des faces qui constituent l'objet tridimensionnel .

IV. Fonctionnement du logiciel

IV.1. Présentation

Notre application est conçue pour décomposer une image 3D en mailles triangulaires elle se présente comme suit :

Premièrement on a la figure d'accueil qui est montrée ci-dessous (figure IV.1)

Figure IV.1 : Accueil de l'application

Comme on peut l'apercevoir dans la figure IV.1 on a le menu qui nous permet d'ouvrir une image 3D d'extension .vtk en cliquant sur le bouton (1) ce qui fait apparaitre la liste des images 3D (2) de notre base de données.

Après que l'utilisateur ait choisi une image il clique sur celle-ci pour l'afficher ce qui donne la figure IV.2.

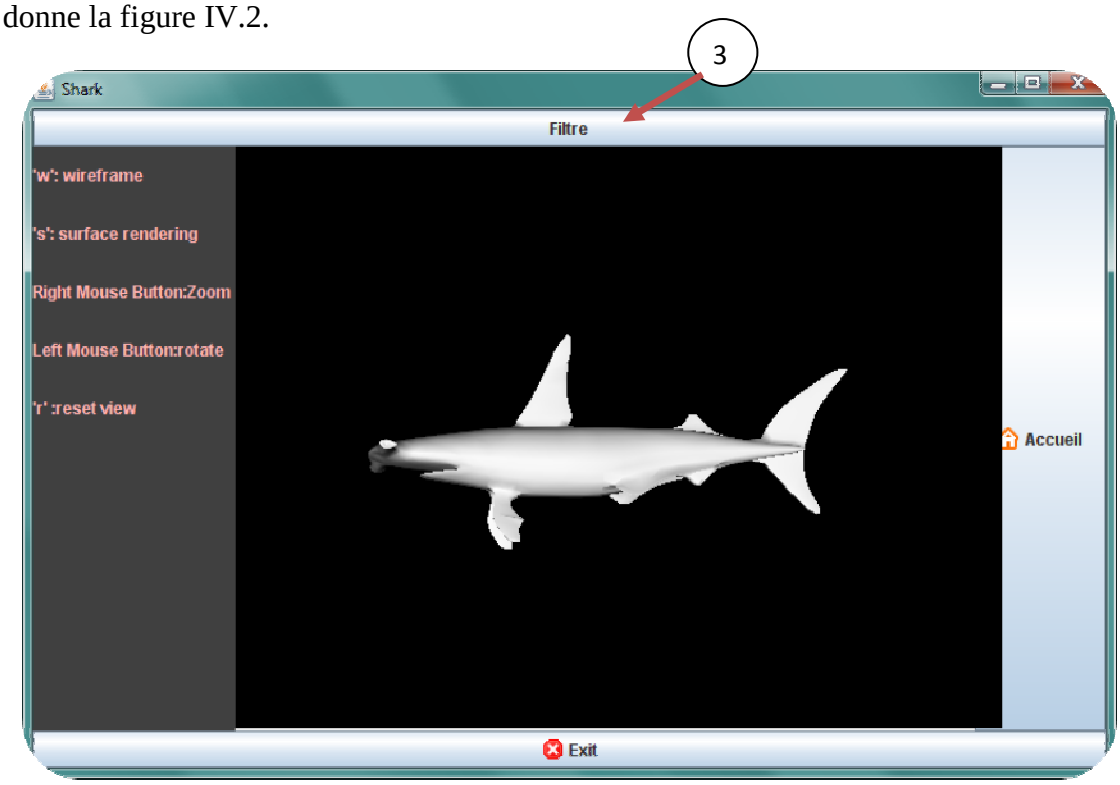


Figure VI.2 : Image 3D .vtk

Après avoir visualisé l'image, on lui applique le filtre triangulaire en cliquant sur le bouton (3) .

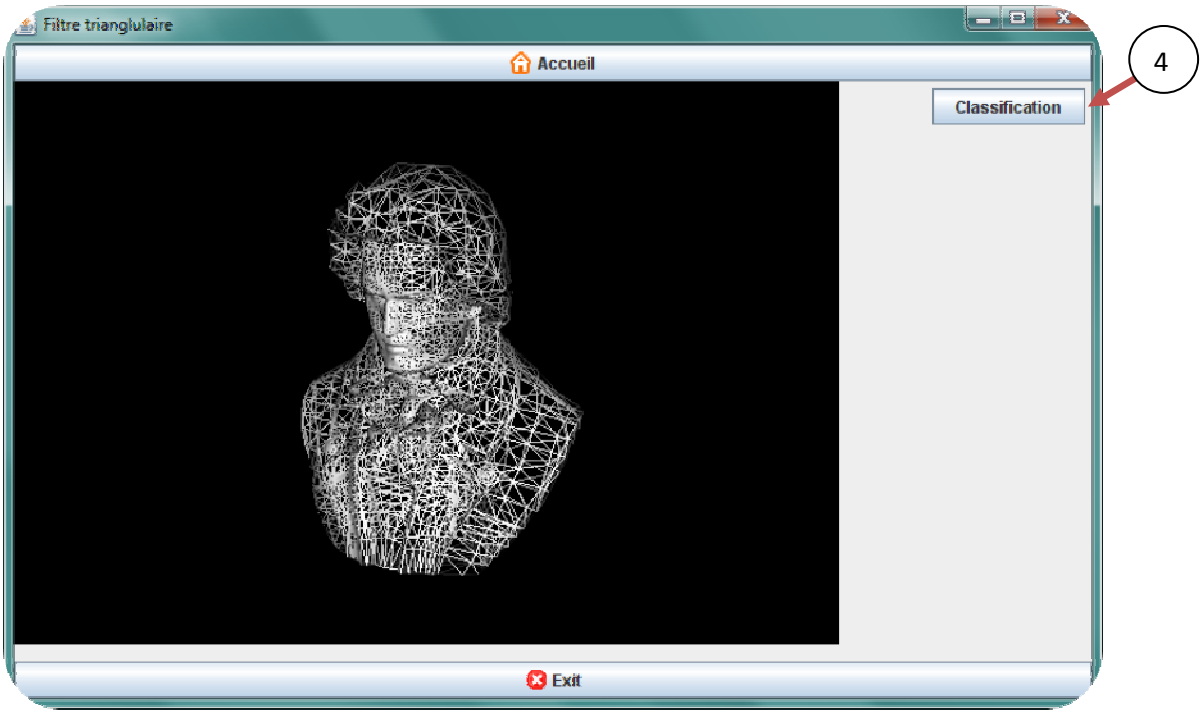


Figure IV .3 : Image 3D décomposée.

Comme on peut le voir sur la figure VI.3 après qu'on est appliqué le filtre sur l'image 3D, on a comme résultat une image décomposée en mailles triangulaires.

L'étape suivante est la classification des régions en cliquant sur le bouton (4)

Qui nous donne la forme suivante (Figure VI.4).

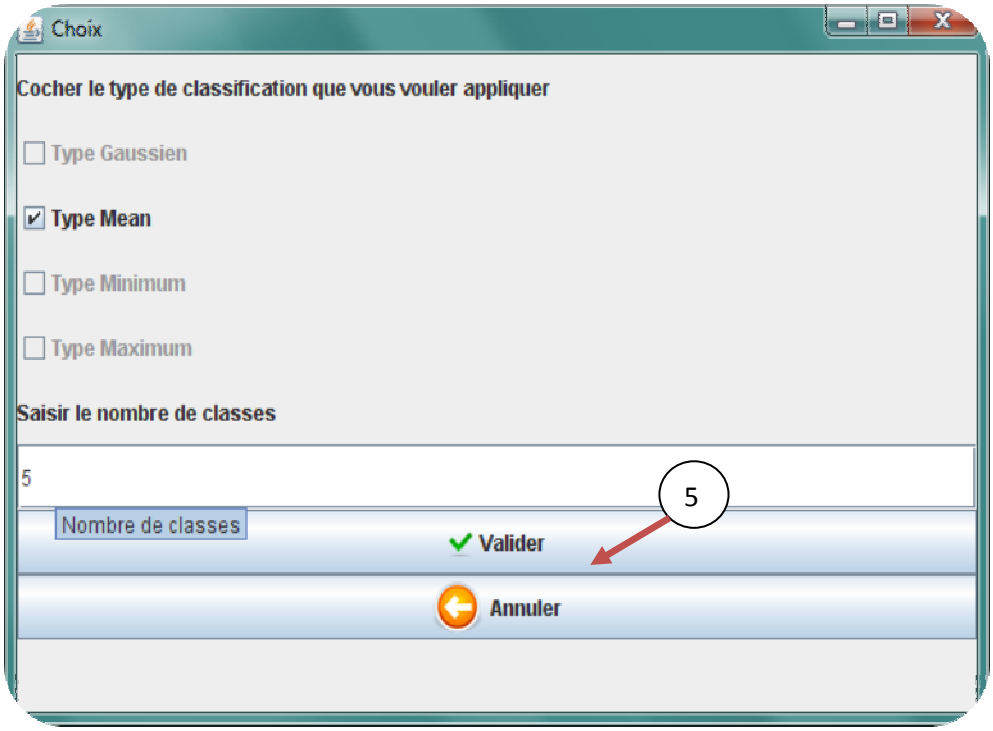


Figure IV.4 : Choix d'algorithme à appliquer

L'utilisateur doit cocher le type d'algorithme à appliquer, une fois son choix est fait, il lui suffit de préciser le nombre de classes et de cliquer sur le bouton (5) et on a le résultat final attendu par l'utilisateur dans la figure (Figure IV.5).

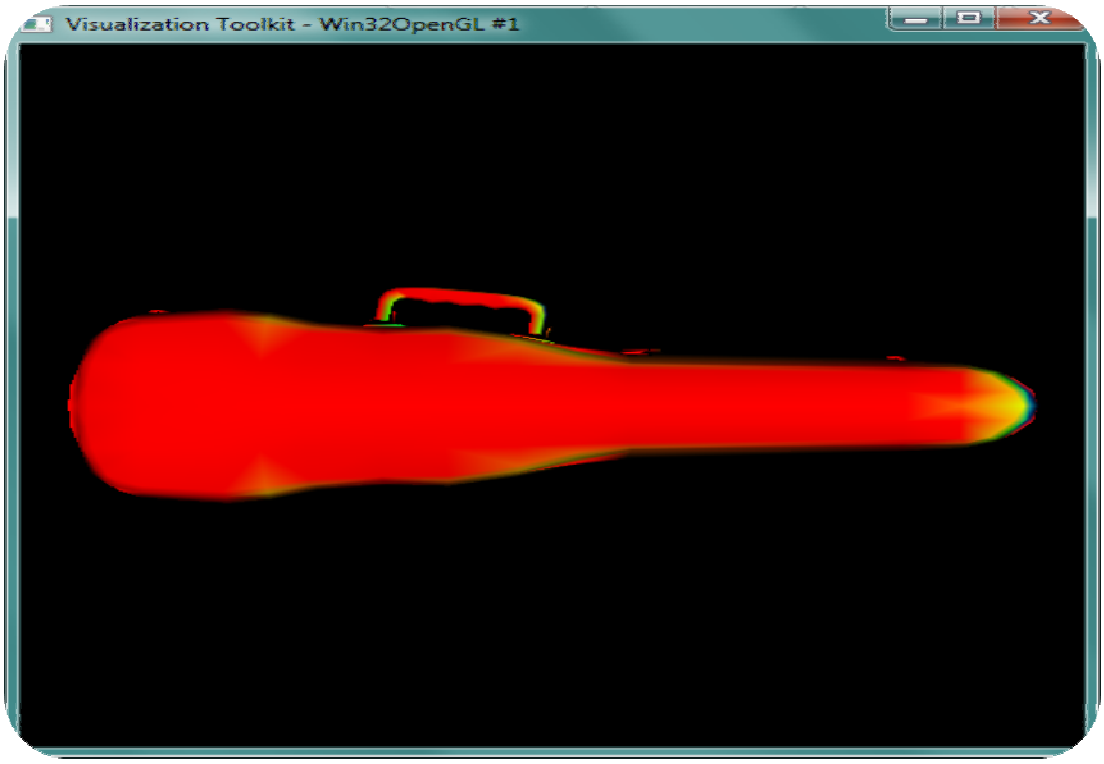


Figure IV.5 : Image 3D résultat

Programme :

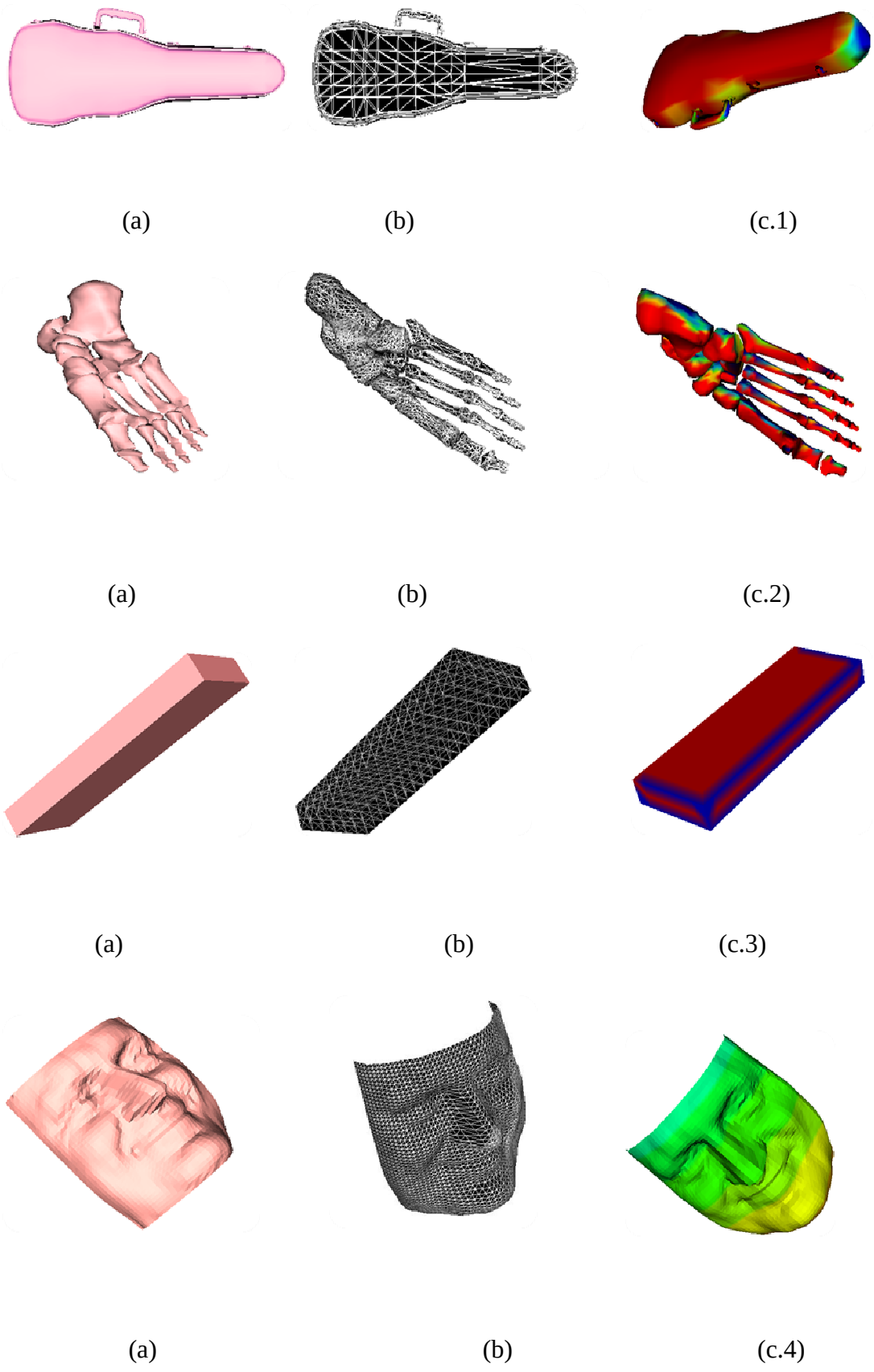
Pour pouvoir ouvrir une image 3D, la librairie VTK nous offre la classe **vtkPolyDataReader**, quand à l'enregistrement de l'image résultat on a utilisé la classe **vtkPolyDataWriter.**

Pour appliquer le filtre triangulaire sur l'image 3D en utilisant la librairie VTK, on a la classe **vtkTriangleFilter**.

Exemple :

```
vtkSphereSource sphere =new vtkSphereSource(); /***Source***/

sphere.SetRadius(5);

sphere.SetThetaResolution (36);
sphere.SetPhiResolution(18);

vtkPolyDataMapper isoMapper=new vtkPolyDataMapper(); /***Mapper***/
isoMapper.SetInput(sphere.GetOutput);
isoMapper.ScalarVisibilityOn;

vtkActor isoActor=new vtkActor() ; /***Actor***/
isoActor.SetMapper(isoMapper) ;

vtkRenderer ren1=new vtkRenderer();
ren1.AddActor(isoActor) ;
ren1.SetBackground(1,1,1) ;

vtkRenderWindow renWin=new vtkRenderWindow(); /***Renderer***/
renWin.AddRenderer(ren1) ;
renWin.SetSize(500,500) ;

vtkRenderWindowInteractor iren= new
vtkRenderWindowInteractor() ;/***Render Window***/

iren.SetRenderWindow(renWin) ;
iren.Initialize ;
```

IV.2 Expérimentation :



(a) (b) (c.1)

(a) (b) (c.2)

(a) (b) (c.3)

(a) (b) (c.4)

Figure IV.6 : (a) Affichage de l'objet 3D, (b) Visualisation de maillage triangulaire, (c) : classification, (c.1): *type minimum*, (c.2): *gaussien*, (c.3) *maximum*, (c.4) : *mean*.

V. Conclusion

Dans ce chapitre nous avons présenté notre application dont le but est la décomposition de maillage triangulaire, en spécifiant les différentes étapes qui constituent notre travail.

Conclusion générale

Conclusion générale

Au cours de ce projet nous avons eu l'occasion de travailler avec les objets 3D, et de mesurer leurs importance dans les applications actuelles, qui visent à donner aux machines la capacité de voire et de détecter afin de pouvoir aider, ou même remplacer l'être humain dans de nombreuses applications, notamment dans le domaine de l'imagerie médicale. De ce fait la décomposition de maillage triangulaire est devenue une étape de prétraitement très importante pour des traitements ultérieurs sur ces objets.

Nous avons présenté dans ce modeste travail, une méthode de décomposition de mailles triangulaires, en se basant sur le critère de la courbure discrète, et en utilisant les triangles pour générer les différentes régions du modèle 3d.

Notre travail s'inscrit dans un objectif plus modeste de segmentation. Un bon nombre d'améliorations sont à envisager :

- Effectuer un algorithme de croissance de régions réunissant les triangles en régions connexes à partir des clusters de courbure des sommets.
- Construction et réduction du graphe d'adjacence des régions afin de fusionner les régions semblables.