École Polytechnique de l'Université de Tours
64, Avenue Jean Portalis
37200 TOURS, FRANCE
Tél. +33 (0)2 47 36 14 14
www.polytech.univ-tours.fr

Département Informatique

Rapport de projet de fin d'études 2014/2015

Exploitation de graphe pour la visualisation d'images 3D multimodales

Encadrant
Jean-Yves RAMEL
jean-yves.ramel@univ-tours.fr

Etudiant
Alexandre LOUBIER
alexandre.loubier@etu.univ-tours.fr

Affiliés
Thierry BROUARD
thierry.brouard@univ-tours.fr
Gaetan GALISOT
gaetangalisot@gmail.com

Université François-Rabelais, Tours

DI5 2014 - 2015

Version du 8 mai 2015

Table des matières

Table des figures

Remerciements

Pour commencer ce rapport, je souhaite remercier un certain nombre de personnes qui sont intervenues durant ce projet de fin d'études.

Tout d'abord, je remercie M. Jean-Yves RAMEL, mon encadrant, pour son aide et sa compréhension tout au long de ce projet. Son assistance a été très appréciable car il a su valider les initiatives que j'ai pu prendre et également me faire comprendre les différents points qui pouvaient être trop flous pour moi.

Ensuite, je souhaite remercier M. Gaetan GALISOT pour les conseils qu'il m'a donné sur l'utilisation souhaitée du logiciel et de s'être procuré pour moi des images de test.

Je souhaite également remercier M. Romain RAVEAUX pour l'aide qu'il m'a apporté sur l'utilisation de la bibliothèque Graphs.

Introduction

Ce document présente le travail réalisé par Alexandre LOUBIER lors de son projet de fin d'études du cursus informatique de l'école d'ingénieur Polytech Tours. Ce projet a été encadré par M. Jean-Yves RAMEL.

Ce rapport contient des détails sur le travail réalisé sur le projet "NeoBrainSeg" dans la continuité des réalisations de M. Amaury BIANCHI lors de son PFE en 2013/2014. Volontairement, ce rapport ne présentera pas de détails trop techniques afin d'être compréhensible par le plus grand public possible, en particulier les personnes qui seraient amenées à reprendre ce projet.

Pour commencer, nous aborderons le contexte du projet avant de parler des points concernant la gestion de projet et le planning de développement. Puis nous parlerons de l'analyse du prototype existant et préciserons les points à modifier et/ou à améliorer. Nous détaillerons ensuite les fonctionnalités et améliorations apportées au logiciel. Enfin, nous finirons en abordant les difficultés rencontrées lors de ce projet de fin d'études et des pistes concernant la poursuite de projet.

Contexte du projet

2.1 Contexte

Depuis plus d'un an, le projet NeuroGeo met en collaboration des chercheurs de l'INRA de Nouzilly travaillant sur des images IRM de cerveaux avec certains membres du laboratoire informatique de Tours pour concevoir des outils d'analyse et d'annotation interactives d'images 3D multimodales permettant de construire des représentations numériques du cerveau (atlas) à partir d'images.

L'objectif visé est de concevoir un logiciel permettant d'isoler de manière itérative les différentes parties du cerveau pour pouvoir ensuite fournir des données anatomiques précises facilement manipulables ou utilisables "informatiquement". L'élaboration d'un tel logiciel a déjà débuté et un prototype permettait de modéliser numériquement un cerveau à la fois sous la forme d'un graphe et sous la forme d'une scène OGRE composées d'objets 3D.

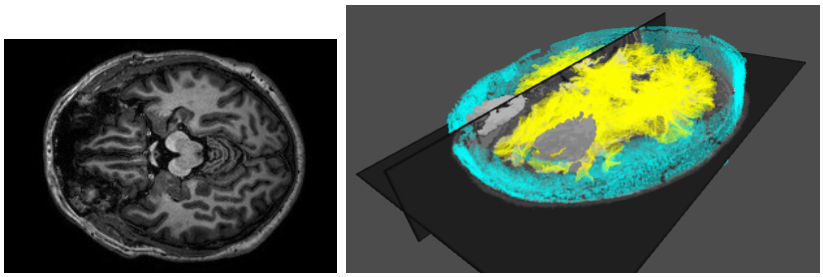


FIGURE 2.1 – Une coupe de cerveau et une représentation en trois dimensions colorée après manipulation

Ce projet s'inscrit également dans le cadre de la thèse de M. Gaetan GALISOT, doctorant au sein du département informatique de Polytech Tours.

2.2 Objectifs

Afin de poursuivre le développement de ce logiciel, le travail à réaliser dans le cadre du PFE à consisté en la réalisation des taches suivantes :

- Analyse de l'existant (techniques utilisées, algorithmes et architecture logicielle)
- Remise en forme de l'architecture logicielle (après re-conception UML)
- Conception et développement des nouvelles fonctionnalités suivantes
- Mise en place des fonctionnalités permettant la visualisation et l'annotation interactive de la scène 3D constituées des différentes régions segmentées.
- Chargement/sauvegarde de la scène courante (graphe + données images et annotations associées)
- Réaliser une documentation technique pour le logiciel.
- Revoir l'Interface Homme-Machine existant afin de la rendre plus interactive et plus intuitive.

2.3 Bases méthodologiques

Afin de poursuivre le travail effectué précédemment sur le logiciel, le développement a été réalisé en C++/CLI avec l'environnement de développement Visual Studio 2012. Les bibliothèques OpenCV, OGRE et Graphs ont notamment été conservées pour ce projet afin de réaliser des opérations complexes (Segmentation, rendu 3D et manipulation de graphes).

Aucun dépôt de sources n'a été disponible au début de ce projet de fin d'étude et aucunes méthodologies particulières n'ont été imposées.

Gestion de projet

3.1 Méthodologie

Pour ce projet, aucune méthodologie n'a été imposée par l'encadrant. Cependant, l'encadrant souhaitant un suivi régulier sur l'avancement du projet et les objectifs susceptibles d'évoluer par l'ajout de fonctionnalités non prévues, nous avons privilégié une méthode itérative avec des ajouts de fonctionnalités unes à unes et validation de la fonctionnalité par l'encadrant. Ainsi, la méthodologie adoptée pour ce projet a été une méthode Agile.

De plus, en début de projet, des réunions avec l'encadrant, M. GALISOT et M. BROUARD ont permis d'identifier les problèmes du prototype existant et mettre en évidence les améliorations possibles. Des contacts fréquent ont également été réalisé avec M. GALISOT afin de présenter le logiciel et de faciliter la reprise du projet qui s'inscrit dans le cadre de sa thèse.

3.2 Logiciels et outils utilisés

3.2.1 Logiciels

Visual Studio 2012

L'environnement de développement Visual Studio 2012 a été utilisé pour poursuivre ce projet. Il permet d'utiliser facilement la bibliothèque .NET et les Windows Forms avec le langage C++/CLI et intègre un grand nombre d'outils facilitant le développement. Pour des raisons inconnues, le projet n'est pas compatible avec la version 2013 de Visual Studio.

Doxygen

Doxygen est un logiciel permettant de générer de la documentation au format HTML à partir des commentaires présents dans le code du projet. Les commentaires doivent respecter le format Doxygen afin d'être reconnus.

Microsoft Project

Microsoft Project a été utilisé afin de réaliser le diagramme de Gantt pour ce projet.

Visual Paradigm

Visual Paradigm est un logiciel permettant de réaliser des diagrammes UML. Le diagramme de classe du projet a été généré à l'aide de ce logiciel.

Redmine

Redmine a été utilisé afin de stocker les sources du projet à l'aide de SVN. Le dépot Redmine est hébergé sur les serveurs de Polytech Tours.

3.2.2 Bibliothèques

OGRE

OGRE (Object-Oriented Graphics Rendering Engine)[6] est un moteur 3D libre, multiplate-forme et open-source orienté scène qui permet à partir d'objets de réaliser un environnement tridimensionnel visualisé en deux dimensions par le biais d'une camera. OGRE ne fournit pas de moteur de rendu 3D car il est en réalité une couche d'abstraction se plaçant au dessus des API OpenGL et Direct3D. OGRE structure ses scènes sous forme de graphe, ce qui nous permet de faciliter l'intégration avec le logiciel. L'utilisation de OGRE dans le logiciel a constitué en la visualisation en trois dimensions des sections et du modèle reconstitué du cerveau. Le wrapper MOGRE (Managed OGRE) est utilisé pour l'intégration de OGRE dans un environnement, .NET dans notre cas une application C++/CLI.

OpenCV

OpenCV[7] est une bibliothèque graphique libre spécialisée dans le traitement d'images en temps réel. Elle est utilisé pour intégrer des algorithmes de segmentations tel que le Kmeans utilisé dans le logiciel.

Graphs

La bibliothèque Graphs est utilisée dans le logiciel pour la gestion et la génération de graphes. Il s'agit d'une bibliothèque C# développée lors d'un PFE (2012/2013) à Polytech Tours destinée à l'exploitation de graphes en reconnaissance des formes.

OpenMP

OpenMP[1] est une bibliothèque permettant de paralléliser des sections de code facilement (voir section 5.10.2). OpenMP est directement intégré à Visual Studio.

FIGURE 3.1 – Logos de Visual Studio 2012, OpenCV, OGRE, OpenMP et Redmine

3.2.3 Gestionnaire de sources

Afin de versionner le projet, un dépôt SVN sur le serveur Redmine de Polytech Tours a été crée. Des livraisons régulières ont été réalisées sur ce dépôt. Le dépôt est accessible à cette adresse :

`http://redmine.polytech.univ-tours.fr/svn/pfevisu3d/trunk`

En plus d'accueillir les sources du projet, le dépot contient les ressources nécessaires afin de pouvoir générer et exécuter le projet (DLL, fichiers de configuration, ressources OGRE, ...). Ceci permettra de faciliter la reprise du projet.

3.3 Planning

Ce projet s'est découpé en trois grandes phases :

— Tout d'abord, une première phase d'analyse du prototype existant qui a amené à la rédaction du cahier de spécifications et au planning initial (voir figure 3.2). Cette phase a occupé la plus grande partie du premier semestre de l'année universitaire 2014/2015
— Ensuite une seconde phase de développement qui a consisté en la ré-architecturisation et de nombreuses corrections sur le prototype existant. Cette phase a occupé la fin du premier semestre.
— Enfin une dernière phase de développement et tests qui a permis de réaliser les nouvelles fonctionnalités définies dans le cahier de spécifications et de nouvelles non prévues initialement.

Les tâches de développement se sont donc organisées de manière itérative dans l'ordre ci dessous. Les numéros de sections associées aux tâches sont précisées entre parenthèses.
— Mise en place de l'affichage en plein écran (5.6) : cette tâche a été réalisée en premier car elle allait dans la continuité des corrections réalisées sur l'IHM.
— Correction de la gestion des modalités (5.2) : cette tâche était la plus prioritaire et a donc été réalisée en deuxième.
— Ajout de l'annulation (5.4) : Tâche non prévue initialement mais qui a été très utile notamment pour le debug de l'application.
— Gestion de fichiers INI
— Sauvegarde/Chargement (5.3) : Autre fonctionnalité très prioritaire pour ajouter de l'utilité au logiciel.
— Annotations en 3D (5.8)
— Synchronisation entre vue 3D et graphe (5.7)
— Lecture de fichiers Nifti (5.9)
— Amélioration des performances (5.10)
— Synchronisation entre coupes et Image 3D : tâche consistant à améliorer la visualisation des images coupes et à afficher les valeurs des voxels pour chacune des modalités.
— Abstraction des algorithmes de segmentation : tâche permettant l'ouverture du logiciel vers de nouveaux algorithmes de segmentation et leur intégration facile.

Nous ne parlerons pas dans cette section des phases de rédactions des différents rapports ou préparation de soutenance. Le planning initial est disponible sur le cahier de spécification réalisé plus tôt pour ce projet.

Le diagramme de Gantt final (figure 3.3) montre le planning réel de développement. Sur ce diagramme, en gris/marron, vous pouvez voir la planification initiale des tâches sur le Gantt initial de référence (figure 3.2).

Note : Deux tâches en particulier ont eu leurs charges surestimées au vue du diagramme de Gantt final (figure 3.3). Pour la première : "Correction de la gestion des modalités", cela était dû au fait que j'avais mal compris les besoins de mon encadrant concernant cette tâche et qui se sont avérées moins complexe que ce que j'avais compris après de plus amples explications. Pour la deuxième : "Amélioration de la synchronisation entre vue 3D et graphe", c'est un changement de méthode par rapport à ce qui était prévu initialement qui a permis de réduire grandement le temps de développement accordé à cette tâche (voir 5.7).

3.4 Diagramme de Gantt

Nom de la tâche	Durée	Début	Fin
Analyse de l'éxistant	**11,86 jours**	**Jeu 18/09/14**	**Jeu 11/12/14**
Lecture des rapports précedent	7 jours	Jeu 18/09/14	Jeu 30/10/14
Analyse de libgraphs	4 jours	Lun 17/11/14	Jeu 11/12/14
Analyse du prototype	9 jours	Jeu 25/09/14	Jeu 20/11/14
Documentation - Redaction	**11 jours**	**Jeu 23/10/14**	**Ven 09/01/15**
Rédaction des spécifications	11 jours	Jeu 23/10/14	Jeu 08/01/15
Préparation du poster	6,14 jours	Dim 14/12/14	Ven 09/01/15
Préparation de la soutenance de mis parcours	6,14 jours	Dim 14/12/14	Ven 09/01/15
Soutenance de mis parcours	0 jour	Ven 09/01/15	Ven 09/01/15
Conception - Réalisation	**55 jours?**	**Jeu 27/11/14**	**Mer 06/05/15**
Mise en place UML et IHM	8 jours	Jeu 27/11/14	Mer 14/01/15
Correction du prototype	7 jours	Jeu 15/01/15	Jeu 29/01/15
Ajout du format Nifti	3 jours?	Mar 03/02/15	Jeu 05/02/15
Mise en place de l'affichage en plein écran	2 jours?	Mar 03/03/15	Mer 04/03/15
Correction de la gestion des modalités	12 jours?	Mar 10/02/15	Jeu 05/03/15
Ajout des annotations	4 jours?	Jeu 05/03/15	Jeu 12/03/15
Sauvegarde/Chargement	8 jours?	Mar 17/03/15	Mer 01/04/15
Synchronisation entre vue 3D et Graphe	9 jours?	Jeu 02/04/15	Mer 22/04/15
Fin du PFE	0 jour	Mer 06/05/15	Mer 06/05/15

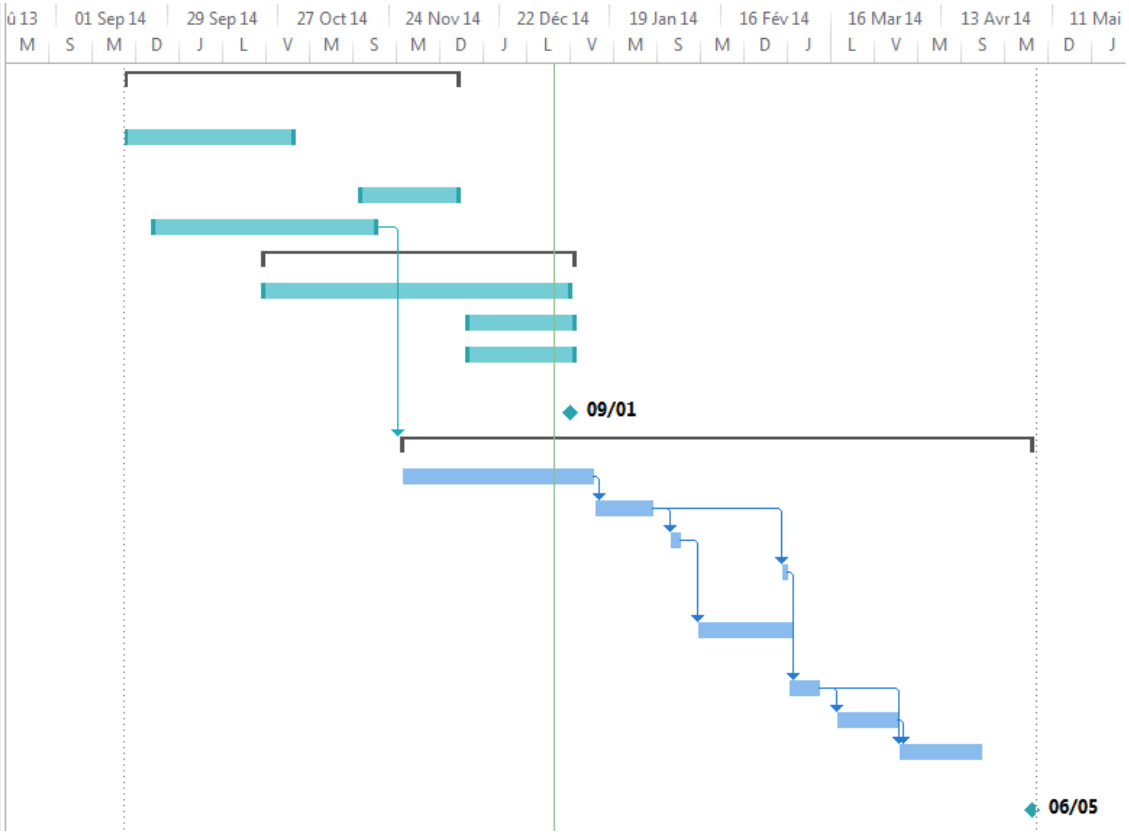
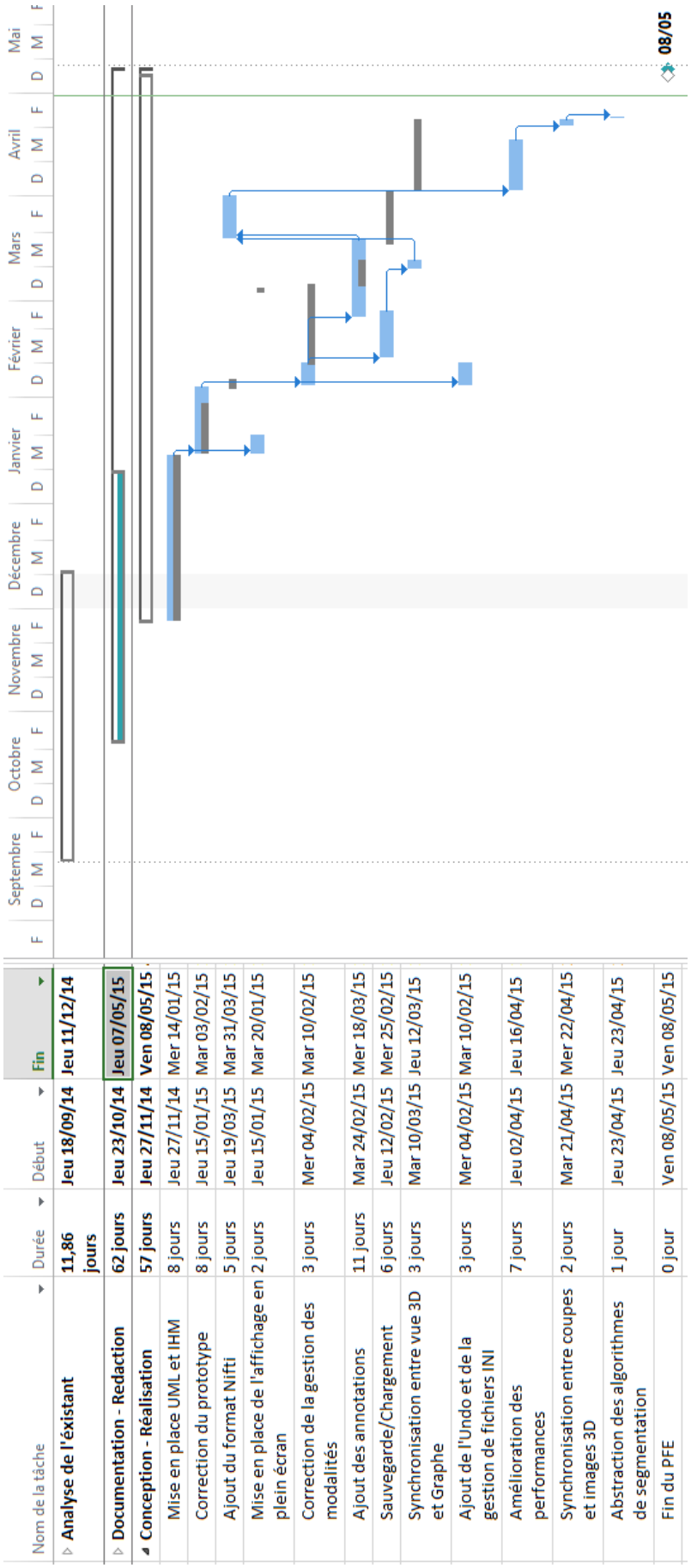


FIGURE 3.2 – Diagramme de Gantt Prévisionnel



Nom de la tâche	Durée	Début	Fin
Analyse de l'existant	11,86 jours	Jeu 18/09/14	Jeu 11/12/14
Documentation - Redaction	62 jours	Jeu 23/10/14	Jeu 07/05/15
Conception - Réalisation	57 jours	Jeu 27/11/14	Ven 08/05/15
Mise en place UML et IHM	8 jours	Jeu 27/11/14	Mer 14/01/15
Correction du prototype	8 jours	Jeu 15/01/15	Mar 03/02/15
Ajout du format Nifti	5 jours	Jeu 19/03/15	Mar 31/03/15
Mise en place de l'affichage en plein écran	2 jours	Jeu 15/01/15	Mar 20/01/15
Correction de la gestion des modalités	3 jours	Mer 04/02/15	Mar 10/02/15
Ajout des annotations	11 jours	Mar 24/02/15	Mer 18/03/15
Sauvegarde/Chargement	6 jours	Jeu 12/02/15	Mer 25/02/15
Synchronisation entre vue 3D et Graphe	3 jours	Mar 10/03/15	Jeu 12/03/15
Ajout de l'Undo et de la gestion de fichiers INI	3 jours	Mer 04/02/15	Mar 10/02/15
Amélioration des performances	7 jours	Jeu 02/04/15	Jeu 16/04/15
Synchronisation entre coupes et images 3D	2 jours	Mar 21/04/15	Mer 22/04/15
Abstraction des algorithmes de segmentation	1 jour	Jeu 23/04/15	Jeu 23/04/15
Fin du PFE	0 jour	Ven 08/05/15	Ven 08/05/15

FIGURE 3.3 – Diagramme de Gantt Final

Analyse de l'existant

4.1 Observations générales

Ce projet est issu d'un certain nombre de projets réalisés au sein de Polytech Tours à savoir deux projets de fins d'études (Amaury BIANCHI en 2013/2014 et Grégoire DECHOUX en 2012/2013) un projet d'ingénierie du logiciel (en 2012/2013) et d'un stage (Erwann CLOAREC en 2012/2013). Il repose également sur des travaux réalisés lors d'une thèse (Ludovic PAULHAC en 2008/2009) et un autre PFE (Alexandre NEVEU en 2011/2012). Une analyse générale du prototype existant est donc essentielle afin de comprendre l'état du prototype sur lequel ce PFE s'est reposé.

L'environnement du logiciel est assez réduit. Aucun accès réseau n'est nécessaire pour le faire fonctionner totalement. L'environnement se limite donc à l'utilisateur, un chercheur de l'INRA, et sa machine, à savoir un ordinateur sous Windows (au minimum Windows XP). Les machines doivent cependant disposer d'une carte graphique compatible avec OGRE et si possible d'une grande quantité de mémoire vive (RAM) afin de pouvoir ouvrir des images plus volumineuses (4Go ou plus).

4.2 Architecture générale du système

Le système s'organise autour de plusieurs composants qui sont :
- Un module de gestion des images 3D multimodales
- Un module de segmentation
- Un module de gestion des graphes
- Un module de gestion des IHM
- Un module de gestion des visualisations en 3 dimensions
- Un classe `Atelier` s'interfaçant avec l'ensemble des modules cités précédemment.

Structurellement, le logiciel ne présente qu'une seule interface graphique composée d'une fenêtre principale et de fenêtres pop up permettant à l'utilisateur d'ouvrir des fichiers ou encore saisir des paramètres de configuration (Kmeans).

La classe `Atelier` sert donc d'interface entre l'IHM et l'ensemble des autres modules de l'application afin de réaliser les traitements attendus par l'utilisateur. On peut noter que la classe `Atelier` et la classe principale d'IHM réalise beaucoup trop de traitements qui devraient être replacés dans les modules correspondants.

Le logiciel comporte trois algorithmes particulièrement complexes. Les deux premiers permettant de convertir l'image de cluster en image de région et l'image de région en graphe ont été expliqués dans les rapports précédents concernant ce projet. En revanche, aucune documentation concernant l'algorithme de génération du modèle 3D à partir de l'image 3D des régions n'était présente. De nombreuses recherches ont été nécessaires afin de trouver la référence pour cet algorithme de meshing [3]

Le diagramme de classe simplifié ci-dessous représente l'architecture du prototype.

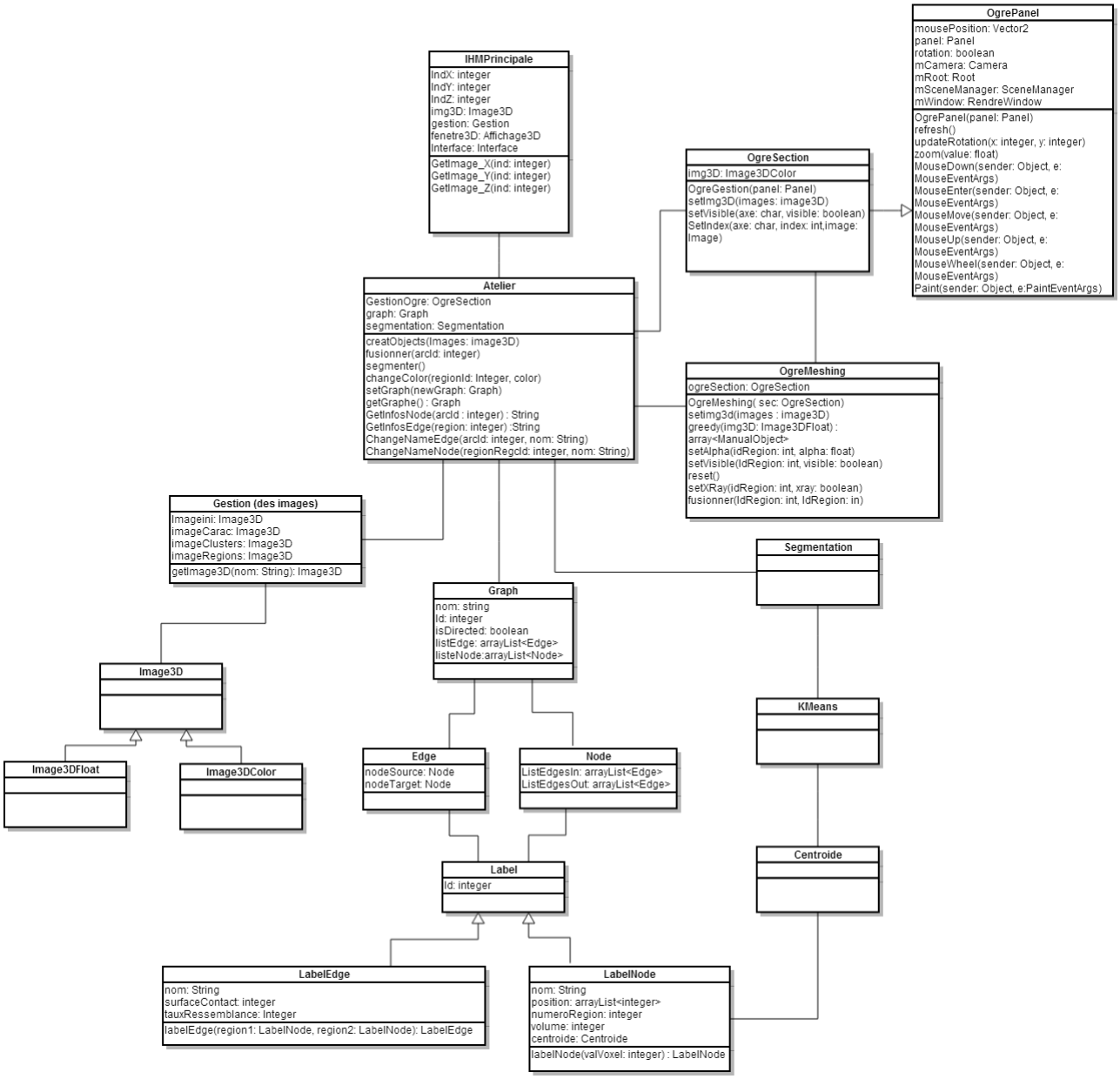


FIGURE 4.1 – Diagramme de classe du prototype simplifié

4.3 Fonctionnalités du système

Le logiciel à réaliser est une application Windows standalone. Son interface se compose uniquement d'une fenêtre principale rassemblant l'ensemble des fonctionnalités de l'application. Des fenêtres de type pop-up peuvent apparaitre afin de notifier l'utilisateur de certaines actions ou de le faire interagir avec certaines fonctionnalités du logiciel.

Le prototype existant permet d'ouvrir et de visualiser des images 3D en niveau de gris issues d'une IRM, de réaliser une segmentation (grâce à l'algorithme du Kmeans) de celles-ci et d'y reconnaitre des régions du cerveau puis de générer un modèle en trois dimensions du cerveau. Ces régions peuvent être renommées et il est possible de changer la coloration de celles-ci ou de les masquer dans la vue en trois dimensions.

Des informations sur les régions sont également affichées lors de la sélection d'une d'entre elles. Les régions identifiées à partir de la segmentation peuvent être segmentée à nouveau afin de les diviser en de nouvelles régions ou bien fusionnées entre elles. Dans la vue en trois dimensions, l'utilisateur peut réaliser une rotation du modèle et zoomer. L'utilisateur ne peut ni sauvegarder, ni charger, ni créer un nouveau projet, il ne peut travailler que sur une image par instance de l'application. En d'autres termes, aucune ré-initialisation de l'application n'est possible.

4.4 Interface graphique

L'IHM principale de cette application est divisée en deux parties. La première est composée d'un panneau de contrôle et de manipulation des données (telle que l'affichage et la modification du graphe des régions et les informations qui les concernent). Cette partie ne constitue qu'une petite partie de l'interface (soit environ un cinquième de sa largeur) cependant beaucoup de place est perdue notamment à cause de l'arborescence de Régions/Arc qui est mal placée et des boutons utilitaires qui prennent trop de places. La deuxième se présente comme une fenêtre de visualisation dans laquelle nous pouvons observer les différentes coupes des images (selon les axes x, y et z) ainsi que la vue en trois dimensions permettant d'afficher une reconstitution du cerveau à partir des résultats des segmentations réalisées sur les images. Cette partie de l'interface est plutôt claire et bien découpée. Elle dispose également d'une barre de menus permettant à l'utilisateur d'utiliser en grande partie les fonctionnalités fournies par le logiciel. Cependant, certains boutons sont activés à certaines étapes de l'utilisation du logiciel alors qu'ils ne le devraient pas (cela risque de provoquer des erreurs) et certains menus ne sont pas vraiment utiles.

Il n'est pas prévu que l'IHM du logiciel soit redimensionnable ou personnalisable par l'utilisateur. La résolution de la fenêtre principale est de 1280x768. Des messages de type pop-up donnent des informations à l'utilisateur sur les validations d'actions ou erreurs générées par l'utilisation du logiciel. Enfin, des fenêtres et barres indiquent la progression des opérations longues telles que le chargement d'images.

L'interface actuelle de l'application est exposée ci dessous :

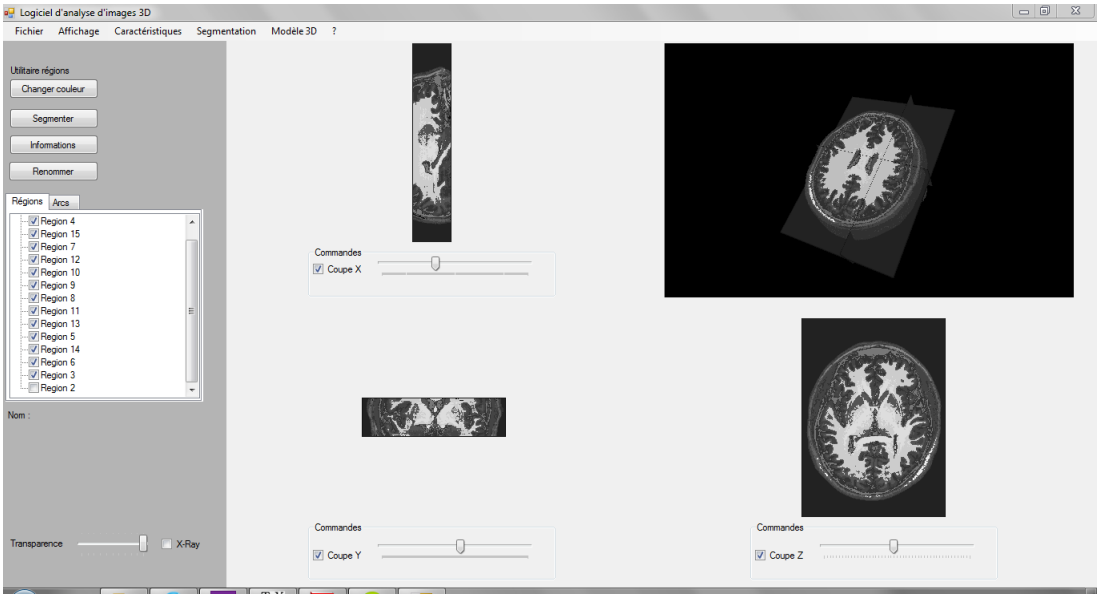


FIGURE 4.2 – Interface du prototype

4.5 Conclusion sur le prototype existant

Le prototype existant est plutôt fonctionnel et permet à l'utilisateur de réaliser la plupart des fonctions attendues pour ce logiciel cependant quelques bugs subsistent et doivent être corrigés notamment au niveau de la segmentation, la fusion, la gestion ou la rotation dans la vue 3D.

Le diagramme de classes précédent doit être revu afin de garantir une meilleure séparation des modules suivant une architecture MVC. L'introduction d'un contrôleur dédié uniquement aux opérations 3D sous OGRE et d'un contrôle réalisant la gestion du graphe sont à réaliser. Ces modifications permettront une meilleure maintenabilité du logiciel.

L'application devra être le plus ergonomique possible afin de permettre une bonne optimisation de l'interface et de son contenu et d'afficher uniquement des informations utiles à l'utilisateur. Les menus et actionneurs de l'IHM devront au maximum gérer les exceptions liés à l'utilisation du logiciels et être désactivés si possible lorsqu'une opération n'est pas réalisable (segmentation d'une image alors qu'aucune modalité n'est disponible par exemple).

Le diagramme d'activité ci-dessous (figure 4.3) permet de présenter le déroulement du fonctionnement souhaité pour l'application et l'ordre dans lequel les différentes actions pourraient être réalisées.

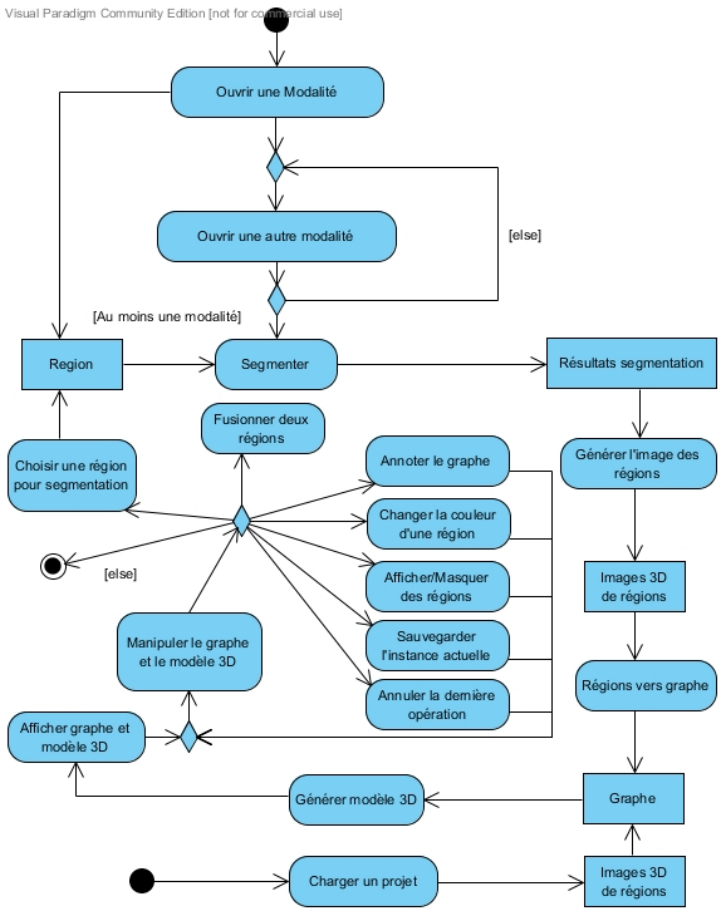


FIGURE 4.3 – Diagramme d'activité

Solutions proposées

5.1 Correction du prototype et restructuration

5.1.1 Refonte partielle de l'interface graphique

Au niveau de l'interface graphique de l'application, quelques changements sont notables. Tout d'abord, le panneau de contrôle (partie gauche de l'interface) a été entièrement modifié afin de l'englober totalement dans un panneaux à 3 onglets.

- Le premier onglet permet de gérer la liste des régions. Un liste des régions est affichée et nous pouvons afficher/masquer ces régions à l'aide de la case à cocher devant le nom de chaque région. Chacune des propriétés des régions sont affichées dans les champs correspondants à la sélection d'une région. Ces champs sont éditables et sont validés lorsque le champ perd le focus ou que l'utilisateur appuie sur la touche Enter . Les boutons de segmentation de la région et de changement de couleur sont toujours présents.
- Le deuxième panneau permet de gérer la liste des arcs. Des champs texte non éditables permettent de visualiser les régions concernées par l'arc sélectionné et de réaliser une fusion en appuyant sur le bouton correspondant.
- Le dernier panneau permet de gérer les modalités. Ce point est détaillé en section 5.2.

Un autre panneau sous le panneau de contrôle permet de visualiser les valeurs des voxels de l'image région et des différentes modalités chargées dans le logiciel sous la position de la souris lorsque celle-ci passe sur les images des coupes.

Le reste des modifications liées à l'interface restent assez mineures et plutôt esthétiques dans l'ensemble (comme la mise en relief des panneaux contenant les images coupes).

La nouvelle interface graphique est visible en figure 5.1 (l'ancienne est visible en figure 4.2).

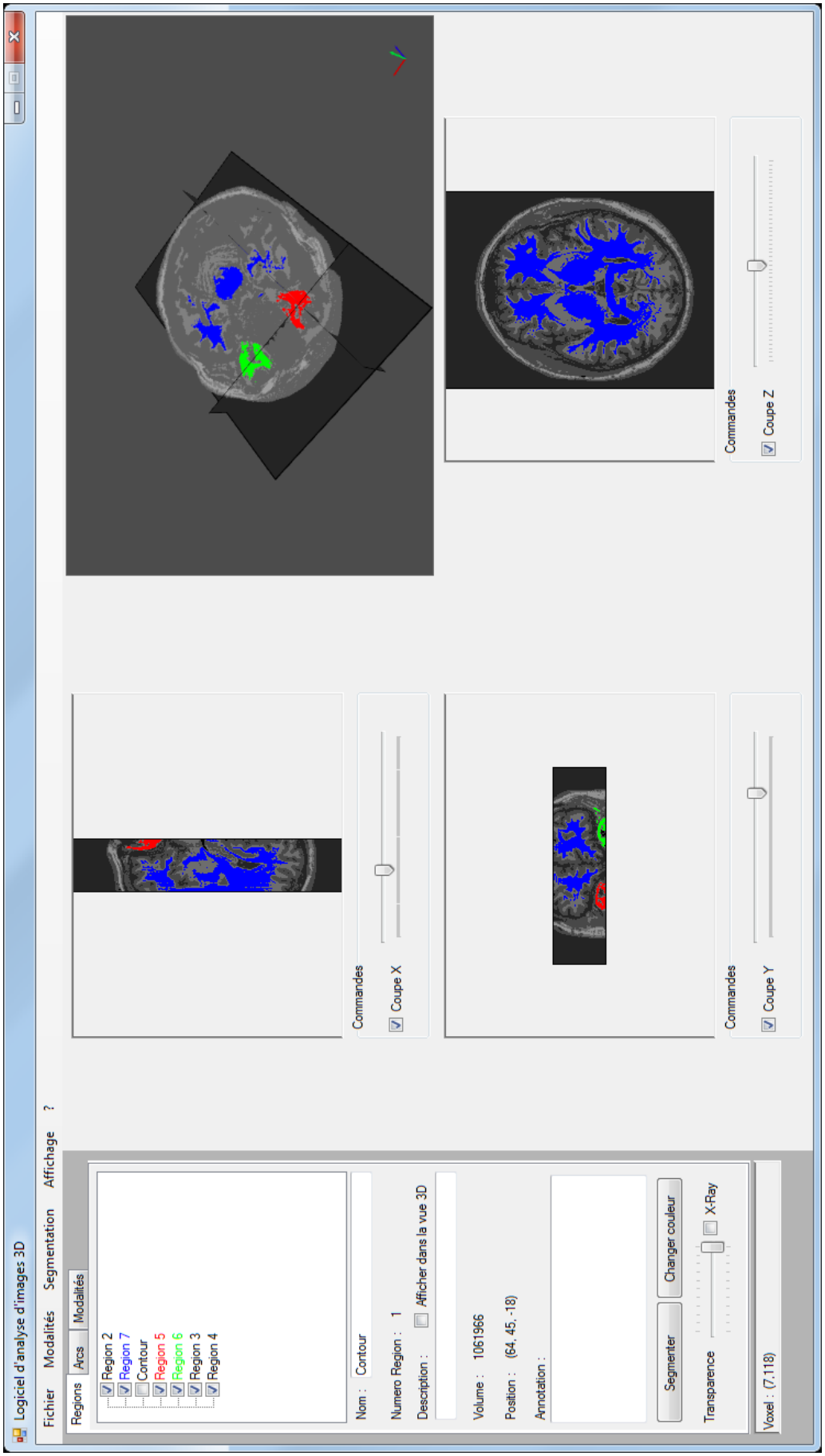


FIGURE 5.1 – La nouvelle interface graphique

5.1.2 Ré-architecturisation partielle

Afin de faciliter le développement, d'ajouter plus de cohérence à l'architecture du logiciel et d'alléger certains modules réalisant trop de traitements différents, celle-ci a été re-découpée. Outre l'ajout de nouvelles classes liées à l'ajout de nouvelle fonctionnalités, certains traitements ont été séparés dans plusieurs contrôleurs. Parmi ces modifications, on peut citer :

- L'ajout d'un contrôleur dédié aux opérations de rendu 3D : `OgreRenderer`.
- L'ajout d'un contrôleur dédié aux opérations de gestion du graphe de régions : `GestionGraph`
- L'abstraction des opérations réalisées par les algorithmes de segmentation (Kmeans, Cmeans) au sein de la classe `AlgoSegmentation`.
- Le remplacement de la classe `Caractéristique` par la classe `Modalité` plus approprié.

Ces modifications sont visibles au sein d'un diagramme UML simplifié présent en figure 5.2. Un diagramme de classe un peu plus complet est présent en annexe B.

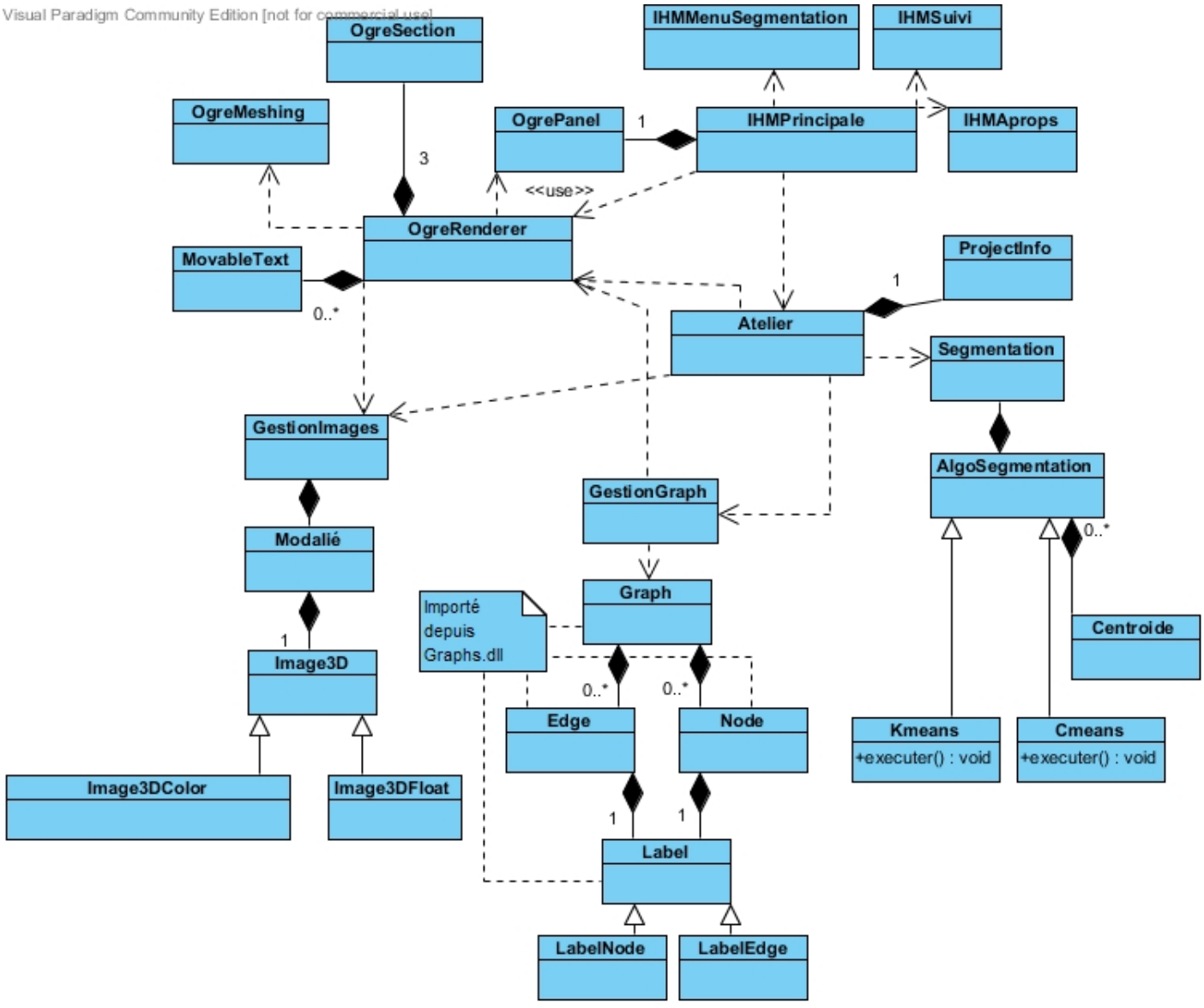


FIGURE 5.2 – Diagramme UML simplifié de l'application

5.1.3 Correction des fonctionnalités existantes

Enfin, j'ai souhaité commencer par corriger certaines fonctionnalités de l'application avant de commencer à en développer de nouvelles afin de partir d'une base de travail plus saine et fonctionnelle.

Ainsi j'ai pu corriger des problèmes liés aux interactions utilisateurs comme l'ajustement des déplacements dans la scène 3D, une meilleure gestion des couleurs, la désactivation d'opérations interdites à certains moments de l'exécution du logiciel ou encore la désactivation de certaines fenêtres pop-up qui gênaient l'utilisation.

De plus, avec l'aide de M. GALISOT, nous avons déterminé les problèmes présents dans les processus de segmentation et de fusion et les avons corrigés. Afin de tester que les algorithmes fonctionnent correctement, des images de tests ont été réalisées (voir Annexe A).

5.2 Correction de la gestion des modalités

Une modalité représente une image 3D qui sera utilisée comme référence afin de réaliser les opérations de segmentation et un nom.

Lors de l'ouverture d'une modalité, l'image est tout d'abord chargée en tant qu'image 3D de couleurs. Puis, grâce à la formule décrite en figure 5.3, nous transformons celle-ci en image3D Float afin qu'elle soit stockée dans la modalité. Ce sont les valeurs flottantes de l'image 3D contenues dans la modalité qui seront utilisées pour la génération de la liste des individus pour l'exécution des algorithmes de segmentation (Kmeans). Il est également demandé à l'utilisateur de rentrer le nom pour cette modalité via une fenêtre pop-up comme le souhaitait l'encadrant.

S'il s'agit de la première ouverture de modalité pour le projet en cours, l'image 3D des régions du cerveau sera générée à partir de cette modalité. Pour qu'une modalité soit compatible avec une image région, il suffit que celle-ci soit de même dimension que l'image région.

$$NDG = 0.299 \times ComposanteRouge + 0.587 \times ComposanteVerte + 0.114 \times ComposanteBleue$$

FIGURE 5.3 – Formule niveaux de gris

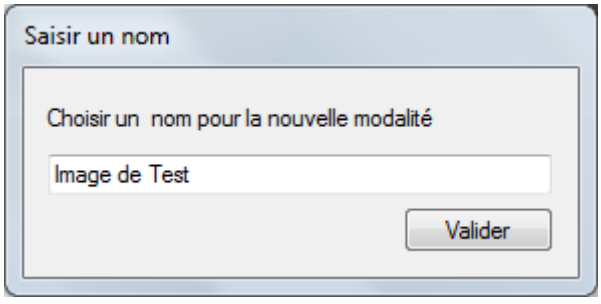


FIGURE 5.4 – Saisie du nom de la modalité

Afin de pouvoir gérer ces modalités au sein du logiciel, nous avons décidé d'intégrer dans le panneau de contrôle un onglet (voir figure 5.5). Afin de ne pas surcharger la consommation mémoire du logiciel, les modalités peuvent être ouvertes et déchargées par l'utilisateur. Le panneau affiche donc la liste des modalités actuellement ouvertes et une liste déroulante permet d'afficher les noms saisis pour l'ensemble des modalités utilisées pour ce projet, afin de conserver une trace de celles-ci même après avoir été déchargées. Enfin, les modalités ne sont pas conservées dans les sauvegardes de projet (voir section 5.3).

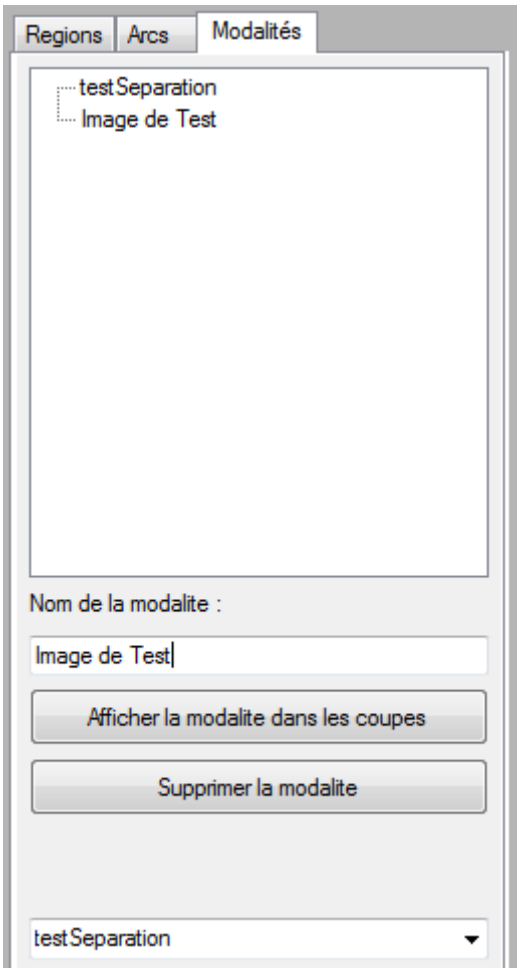


FIGURE 5.5 – Onglet de gestion des modalités

5.3 Enregistrement et Chargement du travail dans l'application

Afin d'ajouter de l'intérêt à l'utilisation de ce logiciel et parce que les opérations principales de celui-ci sont plutôt longues, il était primordial d'ajouter des mécanismes de chargement et de sauvegarde d'un projet.

On considère comme projet, une image 3D de référence et le graphe correspondant à un instant T de l'application. Ces deux structures sont les seules nécessaires afin de conserver le travail entre deux instances de l'application.

Pour le graphe, les opérations de chargement et de sauvegarde sont gérées grâce aux fonctionnalités prévues par la bibliothèque Graphs qui peut manipuler des fichiers `.gxl`. La sauvegarde et le chargement des labels du graphe se font grâce aux fonctions permettant de définir quelles données vont être stockées/lues dans le fichier `.gxl`. De plus, il est nécessaire de préciser lors du chargement du graphe, les labels qui seront utilisés pour les noeuds du graphe (régions) et les arcs.

Pour l'image région, ce sont les couches successives sur l'axe Z de l'image 3D qui sont sauvegardées sous forme de fichiers `.bmp`. Afin de déterminer la couleur de la région, il suffit de prendre le numéro de la région et de le convertir en couleurs RGB 32 bits grâce à un décalage bit à bit (voir figure 5.6). Ainsi, à chaque région est associée une unique couleur qui sera lue lors du chargement pour identifier le numéro de la région.

Des barres de progression ont été ajoutées afin de notifier l'utilisateur de l'avancement des opérations de sauvegarde et chargement.

Conversion numéro région vers couleur

$$ComposanteRouge = (numero \ \& \ 0xFF0000) >> 16;$$

$$ComposanteVerte = (numero \ \& \ 0x00FF00) >> 8;$$

$$ComposanteBleue = (numero \ \& \ 0x0000FF);$$

Conversion couleur vers numéro région

$$numero = ((Rouge \ \& \ 0xFF0000) << 16) + ((Vert \ \& \ 0x00FF00) << 8) + (Bleu \ \& \ 0x0000FF);$$

FIGURE 5.6 – Formule de conversion d'entier en couleur RGB et inversement

5.4 Annulation des opérations

Dans le prototype, certaines actions complexes (telles que la fusion et la segmentation) sont irréversibles et mettent l'application dans un état instable si l'opération échoue. Ainsi, il a été primordial d'incorporer un mécanisme d'annulation. Juste avant la réalisation de ces opérations complexes, le logiciel va sauvegarder automatiquement l'état actuel des données (à savoir le graphe de région et l'image région) pour une utilisation ultérieure.

Une fois les opérations terminées l'utilisateur peut continuer à utiliser les données ou revenir aux données précédentes en appuyant sur le bouton `"Annuler l'opération"` dans le menu `"Segmentation"` et ré-

cupérer ses anciennes données. De plus, si une opération complexe échoue, le logiciel va automatiquement restaurer les données sauvegardées pour récupérer des données de travail stables.

Afin de ne pas surcharger la mémoire, un nombre de données d'annulation maximum est défini dans un fichier `.INI` et lu au démarrage du programme. Si lors d'une sauvegarde de données, le nombre maximum de données est déjà atteint, les données les plus anciennes sont supprimées définitivement.

5.5 Ajout d'un repère 3D

Afin de faciliter la manipulation dans la scène en trois dimensions, nous avons ajouté un repère en trois dimensions permettant de matérialiser l'orientation de l'objet dans la scène.

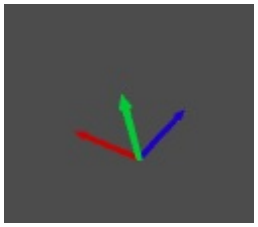


FIGURE 5.7 – Le repère en trois dimensions

Le repère est un objet en trois dimensions présent dans la scène. Il subit les mêmes rotations que le cube de voxel matérialisé dans la scène mais ne subit ni translation ni zoom. Pour permettre ce comportement, nous avons placé le repère très loin dans la scène OGRE et avons créé une camera spécialement pour celui-ci. Le viewport de cette camera s'affiche par dessus celui de la camera principale en bas à droite et permet grâce à un effet de transparence de ne pas masquer cette partie de la vue.

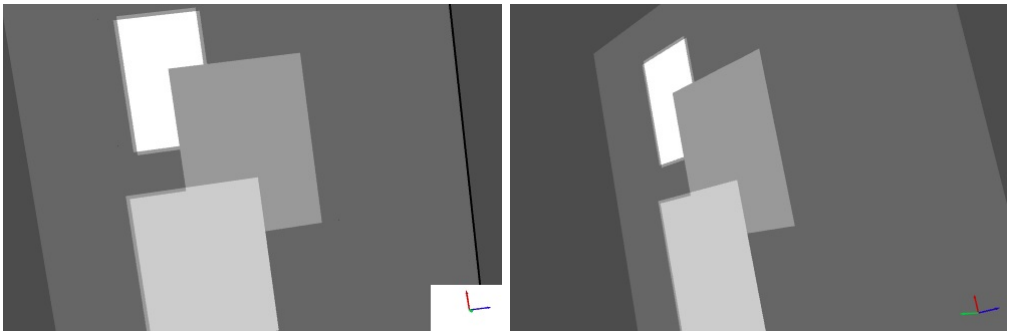


FIGURE 5.8 – Affichage du repère sans et avec transparence

5.6 Affichage en "plein écran"

Dans le logiciel, moins d'un quart de l'interface graphique est originellement consacré à la visualisation du rendu en trois dimensions. Il peut donc s'avérer difficile pour les utilisateurs de visualiser correctement de petites zones à moins de zoomer énormément sur le modèle en trois dimensions.

Afin de palier à ce problème, nous avons mis en place un affichage en "plein écran" dans lequel la vue en trois dimensions occupe la majeure partie de l'interface graphique.

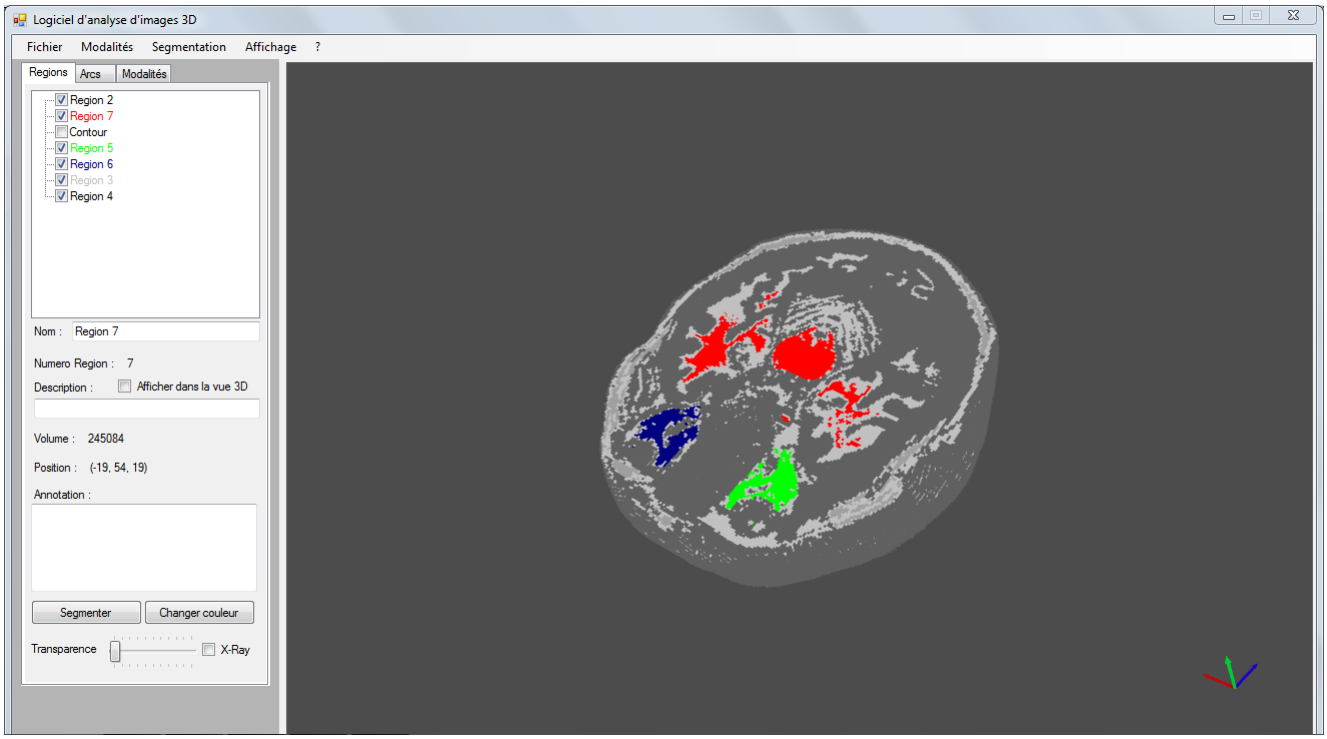


FIGURE 5.9 – L'affiche en "plein écran"

Le passage entre "affichage plein écran" et affichage classique se fait en cliquant dans le bouton correspondant dans le menu de l'application ou lors d'un double-clic dans la vue en trois dimensions.

5.7 Synchronisation entre vue 3D et graphe

Afin d'ajouter de la pertinence à la vue en trois dimensions et de ne pas la limiter à uniquement de l'affichage, nous avons ajouté la possibilité à l'utilisateur de cliquer dans la vue pour sélectionner une région et afficher ses informations dans le panneau d'information des régions. Pour éviter d'entrer en conflit avec les contrôles de position/orientation/zoom déjà réactif aux déplacements de la souris et aux clics dans la scène, il est nécessaire de maintenir le bouton Ctrl enfoncé et de réaliser un clic gauche sur la région souhaitée.

La technique utilisée pour pouvoir déterminer quelle région a été sélectionnée s'appelle le Pixel Perfect Picking[8]. Son principe est d'associer à chaque objet un identifiant unique qui sera retranscrit en une couleur unique dans la scène, puis de regarder la couleur du pixel correspondant à la position du clic de la souris afin d'obtenir l'objet correspondant. Le processus est le suivant :

- Associer un identifiant unique à chaque objet.
- Associer une couleur à chaque identifiant.
- Dessiner la scène en utilisant ces couleurs.
- Récupérer la couleur du pixel sous la souris.
- Récupérer l'identifiant de l'objet.
- Redessiner la scène avec les couleurs normales.

Dans notre cas, à chaque objet en trois dimensions est déjà associé un indice de région, il ne reste plus qu'a associer à chaque région une couleur qui sera utilisée pour le picking. De plus, pour éviter de dessiner dans la vue en trois dimensions et d'observer un changement de couleur très rapide dans la scène, le picking est réalisé sur une texture reproduisant exactement l'état de la scène mais dessinée avec les couleurs générées pour le picking.

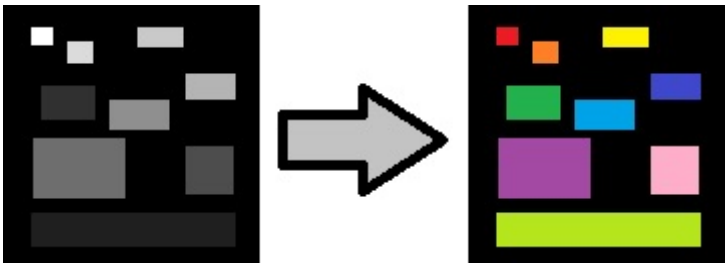


FIGURE 5.10 – Illustration du Pixel Perfect Picking

Sur la figure 5.10 les couleurs sont exagérées, la conversion de couleurs permet d'associer chaque numéro de région à sa couleur RGB 32 Bits correspondante (soit 16777216 couleurs possible, voir figure 5.6). Pour un nombre de régions peu élevé comme celui de la figure 5.10 les différentes couleurs des régions ne pourraient pas être distinguées par l'oeil humain.

Note : Lors de la planification de cette tâche, il était initialement prévu de combiner un lancé de rayon[10] entre la position de la caméra et la position de la souris et une détection de collision pour déterminer la région cliquée. Cependant cette approche s'est avérée plus complexe à mettre en oeuvre de part la complexité des objets gérés dans la scène (pour la plupart concaves) et bien moins performante en terme de vitesse et de considération mémoire.

5.8 Annotations en trois dimensions

Afin de pouvoir ajouter plus d'interactivité à la scène en trois dimensions. Nous avons ajouté la possibilité à l'utilisateur de visualiser le contenu du champ "`Description`" du panneau de contrôle directement dans la scène en trois dimensions. Cet affichage a été rendu possible grâce à la conversion en C++/CLI et à l'utilisation de la classe `MovableTextByBillBoard` récupérée sur le site de OGRE3D [4]. Cette classe permet d'afficher un texte sous forme de Billboard et est directement attachée à un noeud du graphe de scène pour suivre les transformations appliquées aux objets.

Un Billboard [9] est un objet permettant d'afficher une texture 2D de sorte à ce qu'elle fasse toujours face à la camera. Ainsi, le texte à afficher est converti en texture puis inclus dans un Billboard et fait toujours face à la camera.

La partie difficile de l'implémentation de cette fonctionnalité à été de déterminer la position du texte par rapport à l'objet qui lui est attaché. En effet, les objets 3D correspondants aux régions étant générés depuis les résultats des segmentations, et ces objets n'ayant pas de forme convexe, il est très difficile de déterminer un point d'ancrage pour le texte. Ainsi, en accord avec mon encadrant, nous avons décidé de choisir un point aléatoire de l'objet 3D comme point d'ancrage.

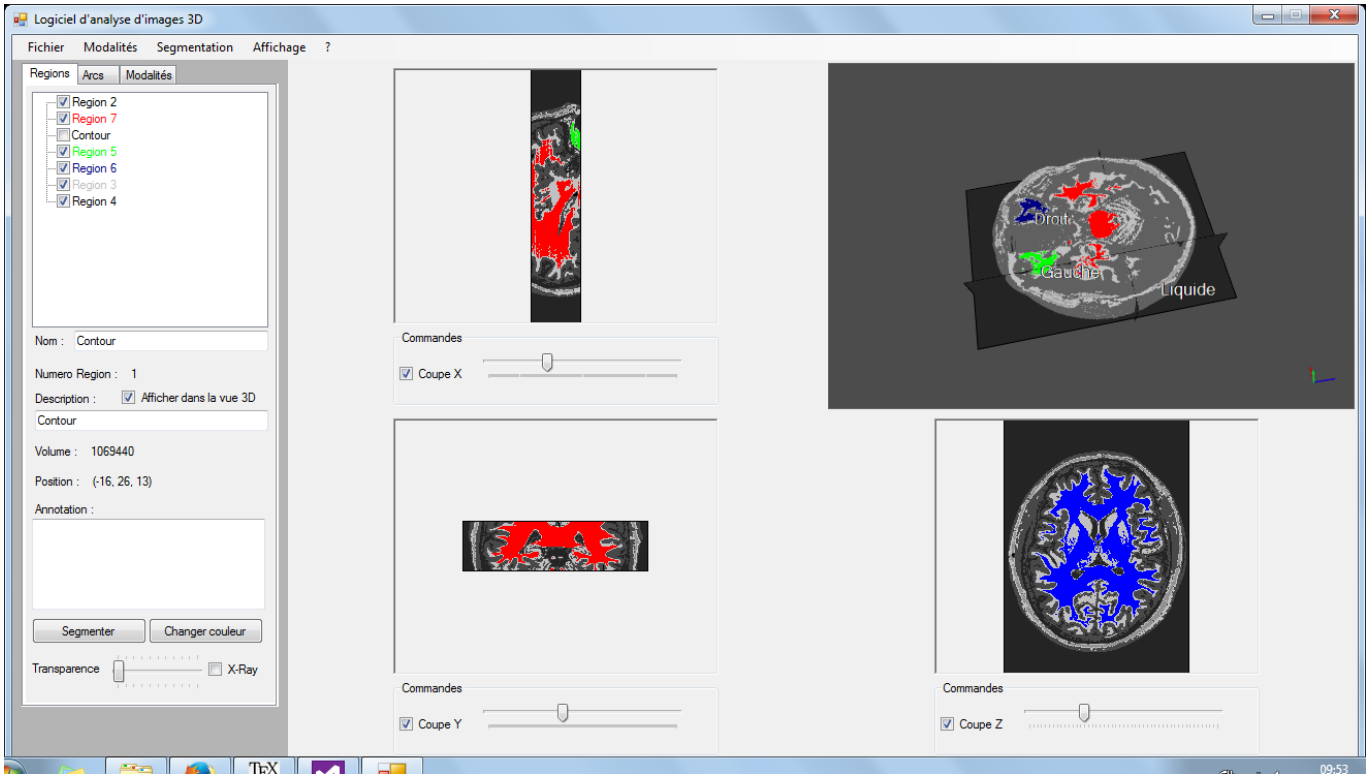


FIGURE 5.11 – Annotations dans la scène en trois dimensions

5.9 Lecture du format d'image Nifti

Enfin, la dernière fonctionnalité planifiée a été la lecture de fichier d'images 3D au format Nifti. Ceci à été réalisée très facilement grâce à l'utilisation de la bibliothèque NiftiLib [5]. Celle-ci contient les entêtes spécifiques au format Nifti ansi qu'un API permettant la lecture et l'enregistrement au format Nifti.

Lors des différents tests sur la lecture de fichier Nifti, j'ai remarqué que plusieurs formats internes au Nifti pouvait exister. Ce format permet de déterminer le nombre de bits utilisés pour encoder un voxel dans l'image. Avec mon encadrant, nous avons choisi de n'en garder que deux au sein du logiciel à savoir :

- DT_UNSIGNED_CHAR : équivaut à 8 bits par voxel. Format implémenté pour pouvoir lire des images de tests trouvées sur internet.
- DT_FLOAT : équivaut à 32 bits par voxel. C'est le format qui a été utilisé pour enregistrer les images Nifti qui nous m'ont été fournies par l'INRA via M. GALISOT.

Afin de pouvoir tester l'exactitude des données chargées lors de l'ouverture de fichiers au format Nifti, j'ai comparé les résultats obtenus avec le rendu proposé par le logiciel ImageJ [2], qui se sont avérés identiques.

L'implémentation de cette fonctionnalité a permis de mettre un problème du logiciel en évidence, à savoir la consommation en mémoire RAM considérable par celui-ci lors de l'ouverture de grosses image 3D (voir section 5.10.1).

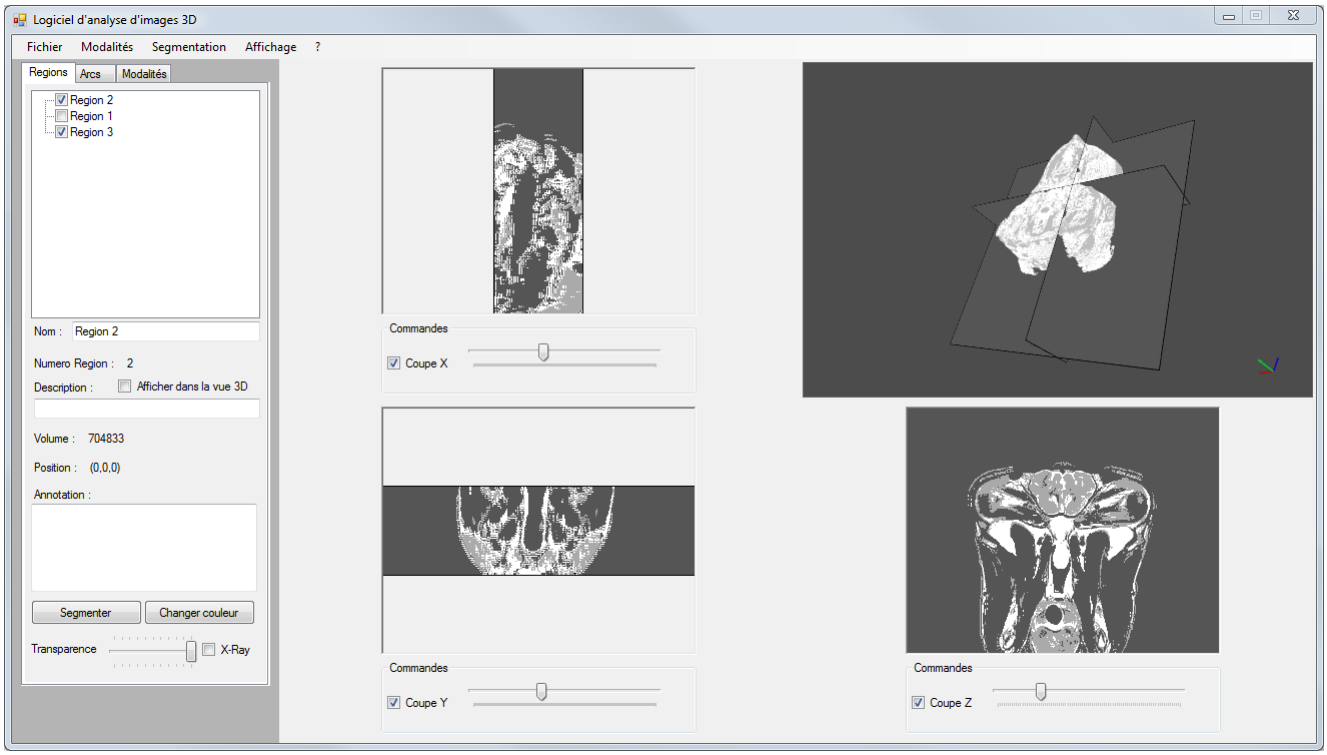


FIGURE 5.12 – Visualisation d'un fichier Nifti fourni par l'INRA

5.10 Amélioration des performances

D'après les rapports précédents, le prototype n'a jamais été testé sur de grosses images 3D. Après avoir réalisé plusieurs tests sur différentes tailles d'image, j'ai remarqué que les performances étaient mauvaises en terme de vitesse et de considération mémoire sur les images de grandes tailles. Les points suivants permettent de mettre en évidence certaines des solutions utilisées pour résoudre les problèmes rencontrés à ce niveau.

5.10.1 Mémoire

Échantillonnage pour la segmentation Lors de la segmentation, les algorithmes utilisés nécessitent des tableaux en entré faisant au minimum 4 fois le nombre total de voxels présents dans l'image 3D ce qui provoque un manque de mémoire vive pour de grandes instances. Pour réduire ce nombre, j'ai réalisé un échantillonnage des données à traiter lorsque la mémoire n'était pas suffisante. Ainsi la segmentation se déroule et un dé-échantillonage est réalisé pour converser l'intégrité des données dans la suite du processus.

Vertex Buffer dans OGRE Par défaut, les Vertex Buffer utilisés par OGRE pour réaliser le rendu 3D ont une taille de 16 bits. Cette taille a dû être augmenté à 32 bits car pour les images de grandes tailles, le nombre de points utilisés excédait cette limite.

Enregistrement des données d'annulation (voir 5.4) sur le disque Initialement, il était prévu d'enregistrer les données d'annulation directement dans la mémoire vive (RAM) de la machine. Cependant, j'ai remarqué que sur les grandes instances, cet enregistrement entrainait une consommation mémoire trop importante. Ainsi, j'ai décidé d'enregistrer (en utilisant les fonctionnalités définies en section 5.3) les données directement sur le disque dur dans un dossier temporaire afin de limiter la consommation mémoire au prix d'un temps plus grand de sauvegarde (environ 1 min pour la sauvegarde sur les grandes instances). Le dossier temporaire est vidé à chaque lancement du programme.

5.10.2 Vitesse

Utilisation d'OpenMP L'utilisation des clauses et directives OpenMP[1] a permis de paralléliser certaines opérations longues telles que des parcours de grands tableaux, les chargements d'images ou les enregistrements. Cependant ces opérations ne peuvent pas être appelées sur des fonctions de manipulations sur les objets Windows Forms ce qui a limité leurs utilisations. L'amélioration de vitesse est proportionnelle au nombre de coeurs disponibles sur la machine (définit le nombre de threads utilisable par OpenMP).

Optimisation d'algorithmes Afin d'améliorer la vitesse, la complexité de certains algorithmes a été amélioré.

Difficultés rencontrées

Le C++/CLI

L'une des difficultés la plus handicapantes pour ce projet a été l'utilisation du C++/CLI. Ce langage permettant d'utiliser l'environnement .NET avec un langage C++ présente un grand nombre d'inconvénient. Sans être exhaustif, des fonctionnalités présentes pour les langages C++ et C# avec Visual Studio ne sont pas disponibles avec ce langage :

- L'outil de refactoring ne fonctionne pas en C++/CLI
- Les arguments de fonction par défaut sont interdit
- Les outils de documentation Intellisense ne fonctionnent pas
- Etc...

En plus de l'absence de fonctionnalités, le C++/CLI semble être un langage peu populaire et peu d'aide concernant ce langage est présente sur les forums et sites spécialisés (notamment pour OGRE et OpenCV). De plus la plupart des classes trouvées sur internet pour réaliser des opérations complexes doivent bien souvent être converties en C++/CLI afin d'être utilisable au sein du projet.

Difficulté de prise en main du projet

La principale difficulté, qui est apparue dès le début du projet, a été la prise en main de celui-ci. Tout d'abord, la récupération du projet et la configuration de celui-ci était complexe puisqu'il était nécessaire d'ajouter la bibliothèque OpenCV au projet dans sa bonne version et de corriger l'ensemble des liens qui étaient erronés. De plus, le projet ne disposait d'aucune documentation mise à part un diagramme UML totalement illisible. Visiblement, aucune conventions de nommages ou de commentaires n'avaient été définies ce qui n'a pas facilité la compréhension du logiciel existant et en particulier son fonctionnement. Pour ce projet, j'ai donc du redéfinir l'ensemble des commentaires avec une norme Doxygen mais une convention de nommage n'a pas pu être appliquée car ceci aurait pris beaucoup trop de temps à cause de l'absence d'outils adapté (voir 6). Une documentation utilisateur très simple a également été réalisée afin d'expliquer l'utilisation du logiciel dans l'état actuel.

Problème de mémoire lors des tests

Enfin, le dernier problème rencontré a été la consommation mémoire phénoménale de l'application. En effet, le logiciel enregistre en mémoire vive l'ensemble des voxels constituant chaque image. De nombreuses optimisations on permis de diminuer grandement la consommation mémoire comme cité en section 5.10.1.

Améliorations possibles - Poursuite de projet

Ajout de nouveaux algorithmes de segmentation

Comme discuté lors des réunions avec l'encadrant, l'amélioration de ce logiciel passerait par l'ajout de nouvelles fonctionnalités liées à des algorithmes de segmentation. Afin de faciliter ce point, une classe abstraite a été extraite des sections de code permettant de réaliser un Kmeans ou un Cmeans afin de faciliter l'implémentation d'un nouvel algorithme.

Fenêtres redimensionnables

Actuellement, les fenêtres affichées par l'application ont une taille fixe. Il serait intéressant de permettre à l'interface de s'adapter automatiquement à la taille de l'écran de l'utilisateur et d'être redimensionnable afin d'optimiser l'expérience utilisateur.

Abandon du C++/CLI

Ce dernier point, à prendre très à la légère, est à considérer étant donné que le C++/CLI constitue un frein au développement de ce logiciel. Mon conseil serait de convertir ce projet en projet Windows Forms en C# ou bien en projet Qt en C++ mais une telle tâche demanderait un effort considérable et beaucoup de temps pour obtenir une application identique. La conversion en C# serait plus facile puisque le projet utilise déjà la bibliothèque .NET mais un projet entièrement en C++ permettrait un meilleur contrôle sur la mémoire consommée par le logiciel.

Conclusion

En prenant du recul sur ce projet, je me suis rendu compte que ce projet était très conséquent et concret. Je tiens à faire remarquer que l'ensemble des personnes ayant travaillé sur ce projet ont réalisé un travail remarquable. Cependant, par le cruel manque de documentation dont ce projet souffrait, il m'a été assez difficile de le prendre en main et de le comprendre comme mon prédécesseur Amaury BIANCHI. J'espère que mes successeurs auront plus de facilité à reprendre ce projet grâce au travail de documentation que j'ai réalisé.

De plus, ce projet comportait des problématiques assez difficiles pour moi à comprendre mais malgré cela, l'ensemble des fonctionnalités prévues ont été réalisées. Nous sommes même allés plus loin que ce qui était prévu initialement.

Enfin, je suis très content d'avoir pu travailler sur un projet mettant en oeuvre de la réalité virtuelle (ici la reconstruction d'un cerveau en 3D et la manipulation de celui-ci dans l'espace) qui est un domaine dont je souhaite faire ma spécialité. J'espère néanmoins que la disparition de la spécialité Réalité Virtuelle en 5^{e}année du cycle ingénieur en informatique de Polytech Tours ne pénalisera pas les étudiants qui seraient amenés à reprendre ce projet.

Images de test

Pour ce projet, certaines images de tests ont été fournies par M. RAMEL et d'autres ont été crées de toute pièces afin de pouvoir vérifier le bon fonctionnement des algorithmes de segmentation et de fusion. Ces images sont des images 3D et son caractérisées par une largeur, une hauteur et une profondeur. Les images de tests que nous avons crées sont identiques en profondeur. Voici les images utilisées à titre de test (seule une seule image de chaque pile est présentée) :

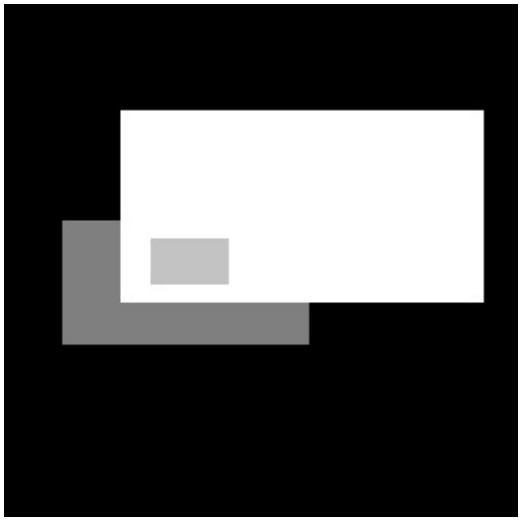


FIGURE A.1 – Image créé mettant en évidence l'inclusion de région.

FIGURE A.2 – Image créé mettant en évidence l'espacement entre régions.

FIGURE A.3 – Image créé mettant en évidence la séparation entre régions connexes.

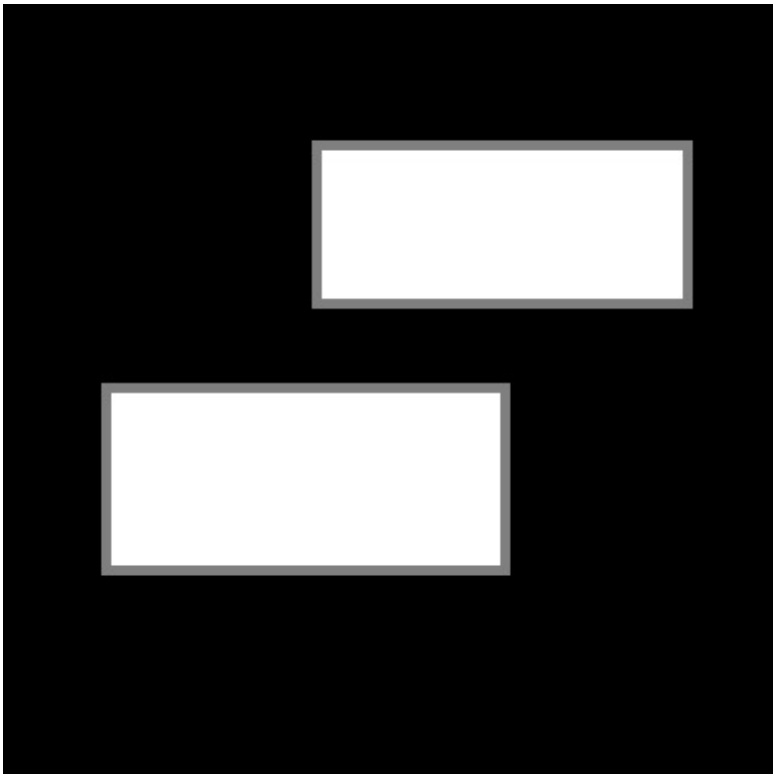


FIGURE A.4 – Image créé de grande taille (384 × 384 × 210) permettant de tester le logiciel sur de grandes dimensions.

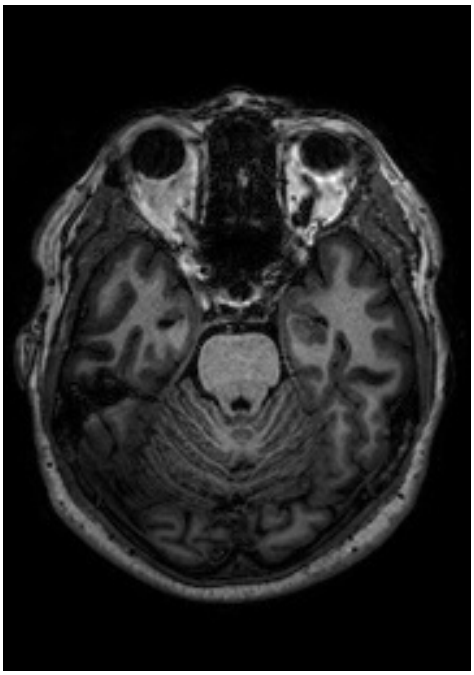


FIGURE A.5 – IRM de cerveau humain.

FIGURE A.6 – IRM de cerveau de brebis.

Diagramme UML avec fonctionnalités importantes

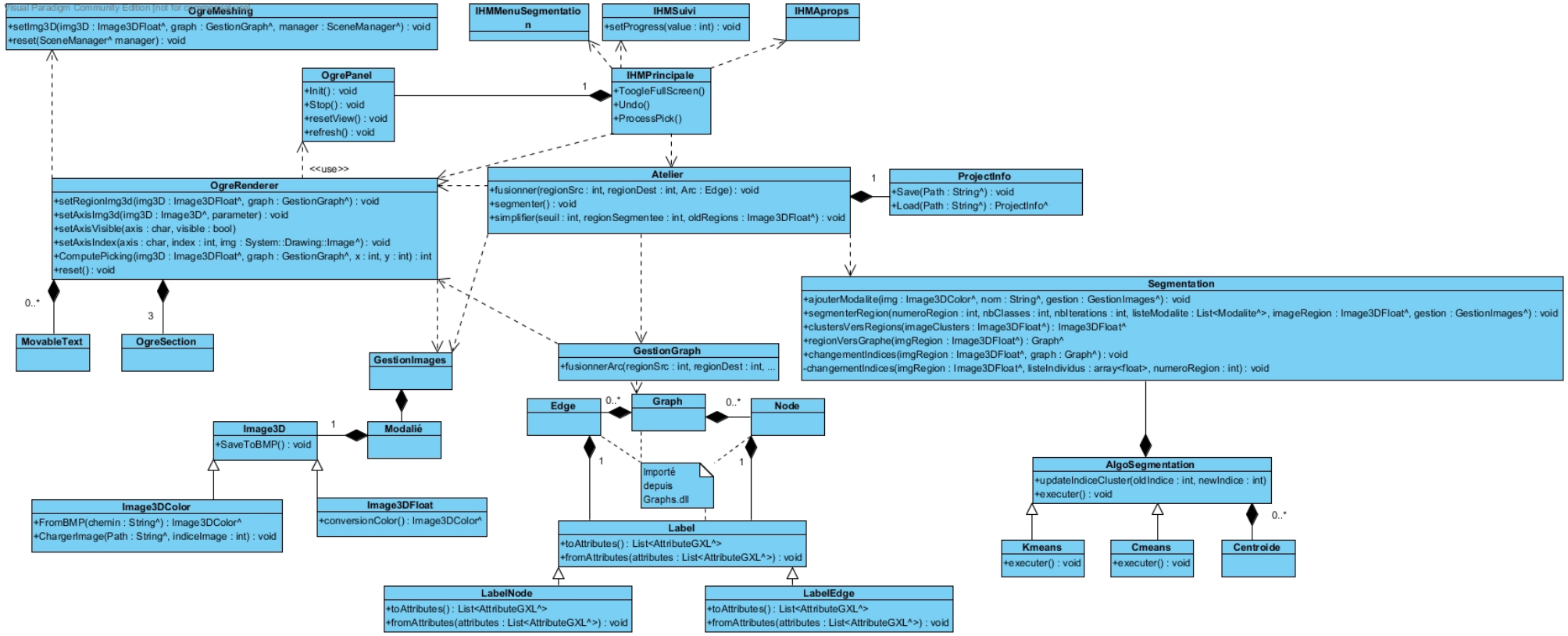


FIGURE B.1 – Diagramme UML avec fonctionnalités publiques importantes

Bibliographie

[1] OpenMP dans Visual C++. https://msdn.microsoft.com/fr-fr/library/tt15eb9t.aspx.

[2] ImageJ. http://imagej.nih.gov/ij/.

[3] 0 FPS Meshing in a minecraft game. http://0fps.net/2012/06/30/meshing-in-a-minecraft-game/.

[4] Classe MovableTextByBillboard. http://www.ogre3d.org/tikiwiki/tiki-index.php?page=MOGRE+MovableText+by+Billboards.

[5] NiftiLib. http://niftilib.sourceforge.net/.

[6] OGRE 3D. http://www.ogre3d.org.

[7] OpenCV. http://opencv.org/.

[8] Color picking tutorial. http://www.lighthouse3d.com/opengl/picking/index.php?color1.

[9] OGRE Wiki Billboard. http://www.ogre3d.org/tikiwiki/tiki-index.php?page=-billboard.

[10] OGRE Wiki Raycasting to the polygon level. http://www.ogre3d.org/tikiwiki/tiki-index.php?page=Raycasting+to+the+polygon+level.

Exploitation de graphe pour la visualisation d'images 3D multimodales

Département Informatique

Rapport de projet de fin d'études 2014/2015

Résumé : Ce document a pour but de présenter la poursuite du projet NeoBrainSeg qui consiste en un logiciel de manipulation et de visualisation d'images 3D multimodales sous forme de graphe. Ce PFE a permis d'améliorer le logiciel en corrigeant certaines fonctionnalités comme la segmentation ou la fusion de régions ou d'en ajouter de nouvelles comme l'ouverture de fichiers Nifti, la sauvegarde et le chargement de projet ou encore l'annulation d'opération. La documentation pour ce projet a du être également être mis à jour car elle était très peu présente.

Mots clefs : Voxel, Image 3D, Multimodale, Cerveau, OGRE, OpenCV, Graphe

Abstract: The purpose of this document is to present the pursuit of the project NeoBrainSeg which consist in a manipulation and visualisation of 3D multimodal images software. This final project study permitted to upgrade the software by correcting some functionalities like segmentation or fusion and to add some like the openning of Nifti files, save and load functionalities or undo functionality. The documentation for this project have also been updated because it was nearly absent.

Keywords: Voxel, 3D Image, Multimodal, Brain, OGRE, OpenCV, Graph

---

Encadrant
Jean-Yves RAMEL
jean-yves.ramel@univ-tours.fr

Etudiant
Alexandre LOUBIER
alexandre.loubier@etu.univ-tours.fr

Affiliés
Thierry BROUARD
thierry.brouard@univ-tours.fr
Gaetan GALISOT
gaetangalisot@gmail.com

DI5 2014 - 2015

Université François-Rabelais, Tours