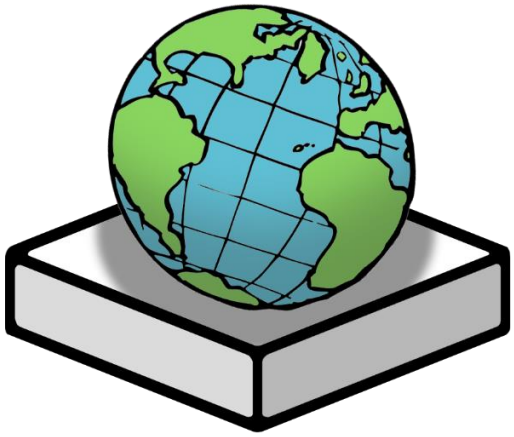
Amélioration d'un logiciel de génération de maillages 3D pour l'impression de cartes tactiles



Rapport de stage

Pierre CHARLES

Master 1 - Technologies Biomédicales

Faculté de médecine – Clermont-Ferrand

L.I.M.O.S

Tuteur : M. Jean-Marie FAVREAU

2016/2017

J’autorise la diffusion de mon rapport de stage.

Les mots suivis d’un astérisque (*) sont à retrouver dans le lexique du rapport.

REMERCIEMENTS

Je tiens tout d'abord à remercier mon tuteur de stage, Monsieur Jean-Marie FAVREAU, maître de conférences spécialiste en géométrie algorithmique, pour m'avoir accepté au sein du Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes et avoir mis à disposition tous les moyens nécessaires au bon déroulement de mon stage.

J'adresse mes remerciements à Madame Sandrine RAMERY, Monsieur Cédric TEVENAIS, Monsieur Olivier PARADIS et Monsieur Pierre MOREAU pour toutes les informations qu'ils ont pu me fournir, pour leur implication dans le projet et pour avoir été à mon écoute pour toutes les questions relatives au projet durant les phases d'analyse et de conception.

Je tiens à remercier particulièrement Monsieur Jérôme ROLLAND, Monsieur Benjamin BOUT et Monsieur Nicolas DESFORGES-DESAMIN, également stagiaire à mes côtés, pour leurs aides, leurs bonnes humeurs et pour avoir su me mettre rapidement à l'aise dans de bonnes conditions de travail.

Je remercie Monsieur Laurent SARRY, chercheur spécialiste en traitement d'images médicales et responsable de ma formation, pour ses disponibilités et ses conseils.

SOMMAIRE

INTRODUCTION

Étant étudiant en master 1 technologies biomédicales à la faculté de médecine de Clermont-Ferrand, j'ai choisi de réaliser mon stage au sein du Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes (LIMOS). En optant pour ce laboratoire, mon objectif était d'intégrer le projet initié en octobre 2016 par Monsieur Guillaume TOUYA, ingénieur des travaux géographiques et cartographiques de l'État et Monsieur Jean-Marie Favreau.

Ce stage se place dans le contexte d'un partenariat naissant entre une équipe de recherche de l'Institut national de l'information géographique et forestière (IGN) et le thème G4 du LIMOS. Il consiste à concevoir une chaîne de traitement logicielle permettant de gérer automatiquement des cartes imprimables en 3D et permettant à des personnes atteintes de malvoyances de consulter le plan d'un quartier. Plusieurs étapes de cette chaîne de traitement sont en cours de réalisation.

Le présent stage consiste à consolider le code existant du logiciel permettant de fabriquer un maillage 3D prêt à l'impression à partir d'une carte en niveau de gris, puis à identifier et implémenter des améliorations du maillage afin de faciliter son impression et améliorer l'ergonomie de l'objet 3D résultant pour des utilisateurs non-voyants.

Après vous avoir présenté le laboratoire qui m'a accueilli et le projet qu'il développe, j'aborderai les objectifs des missions, les contraintes posées ainsi que mon organisation concernant le sujet qui m'a été confié, à savoir, la poursuite et l'amélioration d'un logiciel d'impression 3D de cartes tactiles des étudiants de l'Institut Universitaire et Technologique en Informatique. Je détaillerai ensuite les améliorations et les fonctionnalités ajoutées à l'application. Enfin, je conclurai en dressant un bilan complet du projet.

I) PRÉSENTATION DU STAGE

A) A propos du laboratoire

Le Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes est une unité mixte de recherche en informatique, et plus généralement en Sciences et Technologies de l'Information et de la Communication (STIC). Le LIMOS est principalement rattachée à l'Institut des Sciences de l'Information et de façon secondaire à l'Institut des Sciences de l'Ingénierie et des Systèmes (INSIS). Il a pour tutelles académiques l'Université Clermont Auvergne (UCA) et l'Ecole Nationale Supérieure des Mines de Saint-Etienne (EMSE), et comme établissement partenaire SIGMA Clermont.

Figure 1 : Logo du laboratoire LIMOS

Le LIMOS collabore avec la recherche et le développement de plusieurs entreprises telles que Michelin dans le domaine de la modélisation et de la simulation de pneumatique ou Yansys, une petite entreprise spécialisée dans le domaine des logiciels médicaux. Le positionnement scientifique du LIMOS est centré autour de l'Informatique, la Modélisation et l'Optimisation des Systèmes Organisationnels et Vivants.

Les effectifs du LIMOS sont de 176 au total composés de 81 enseignants-chercheurs, 5 chercheurs, 2 ingénieurs de recherche, 2 ingénieurs d'études, 4 techniciens/administratifs, 1 post-doctorant et de 80 doctorants.

B) Contexte et intérêts du sujet

En partenariat avec un laboratoire de recherche de l'IGN, un travail de génération automatique de cartes tactiles pour les non-voyants et les déficients visuels est en cours de conception. Il s'agit de permettre à ces personnes d'utiliser une carte tactile afin de prendre connaissance de la configuration d'un quartier (carrefours, passages piétons, feux, obstacles sur le trottoir ou encore arrêts de bus ou de tramway).

Pour ce projet, j'ai intégré une équipe composée de trois autres stagiaires. Ainsi, chacun de nous travaillait sur un aspect différent d'une chaine de traitement du projet. Mon travail se place à la fin de celle-ci et se concentre essentiellement sur la génération d'objets 3D. L'objectif du présent projet consiste à améliorer et développer des fonctionnalités d'un logiciel qui va générer le ou les maillages* prévus pour l'impression 3D* volumique de la carte. La figure suivante présente les différents aspects de cette chaîne de traitement.

Figure 2 : Schéma présentant les différents aspects de la chaîne de traitement

Cette partie du projet consiste à générer des fichiers 3D à partir d'une image en niveaux de gris* et plus particulièrement à partir d'une application permettant de transformer une image classique en un maillage 3D imprimable. Les images à transformer seront plus spécifiquement des plans de quartier ou de ville en niveaux de gris. Des données extraites d'OpenStreetMap (OSM) permettront à l'IGN de concevoir ces images. Elles comportent des informations sur le lieu en aplats de noir. Les pixels* en niveaux de gris sur l'image seront transformés en une hauteur 3D permettant, une fois imprimée, aux déficients visuels d'avoir un plan tactile avec des informations en braille. Le résultat final de ce projet doit pouvoir proposer un accès à des cartes tactiles à des particuliers ou des associations souhaitant imprimer une carte d'un lieu qu'ils veulent découvrir.

Plusieurs aspects ont été considérés dans le travail des anciens étudiants. Il existe des normes concernant la hauteur et les dimensions de cellule de braille. Nous avons notamment échangé avec l'association Braille & Culture qui nous a renseigné sur ces normes lors de diverses réunions.

C) Analyse de l'existant

L'interface graphique

L'interface graphique de l'application a été réalisée en Java FX*. Les anciens étudiants ont choisi de faire une interface assez simple d'utilisation pour que l'utilisateur puisse comprendre rapidement les étapes de la génération du maillage. La figure suivante représente des captures d'écrans des différentes fenêtres de l'application existante :

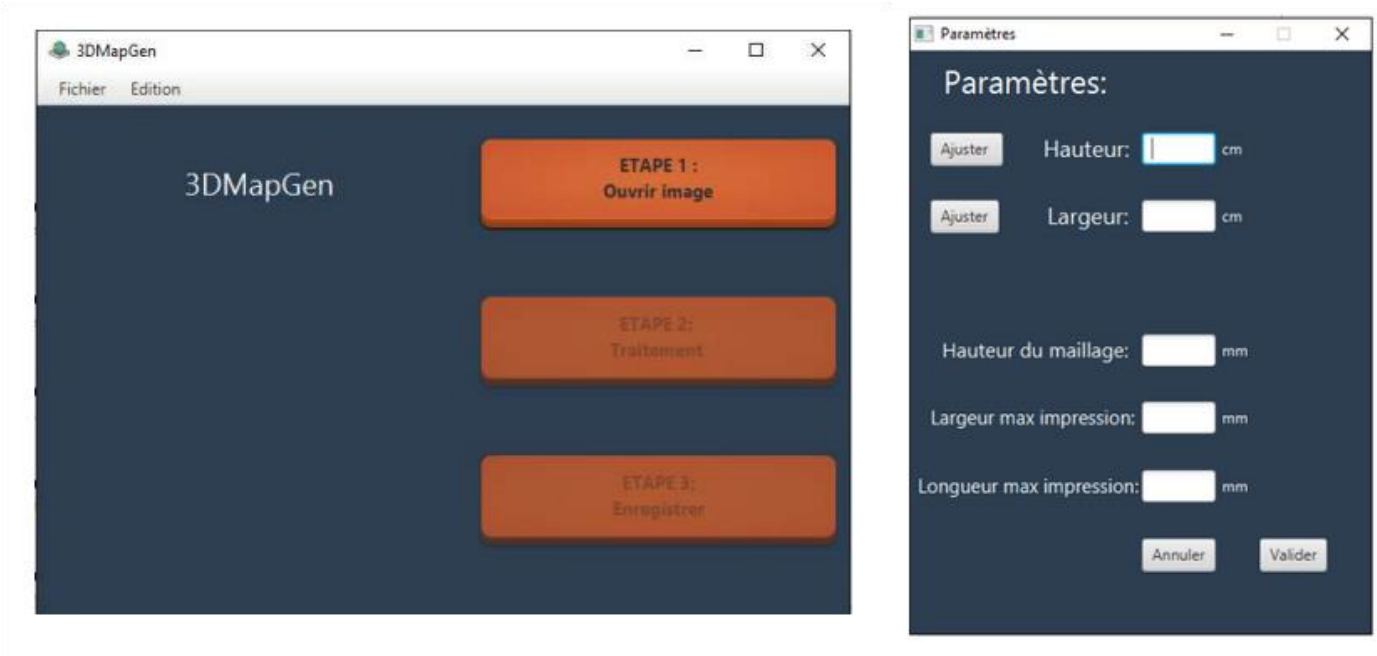


Figure 3 : Captures d'écrans des différentes fenêtres de l'application

Plusieurs étapes permettent la génération du maillage 3D :

- Étape 1: l'utilisateur peut choisir une image, puis choisir des paramètres.
- Étape 2: une fois les paramètres indiqués, le traitement peut être lancé.
- Étape 3: l'utilisateur peut maintenant exporter les fichiers de l'objet 3D.

Les paramètres permettent de définir des variables* pour le traitement de l'image selon la taille de son imprimante et de ces préférences au niveau de la taille de la pièce finale. La partie gauche de l'interface est utilisée pour visionner l'image à traiter. De plus, l'application possède un menu d'édition permettant de changer de thème de couleurs. Un thème avec des couleurs plus sombres et un thème en noir et blanc et des polices d'écriture plus grosses pour être plus facilement utilisable pour une personne malvoyante.

Génération du maillage

Un maillage est un ensemble de formes géométriques disposées de manière à modéliser des objets. Il est constitué de sommets, connectés les uns aux autres par des faces. Lorsque toutes les faces sont des triangles, on parle de maillage triangulaire (Trimesh). Ainsi, le maillage actuellement généré est composé uniquement de Trimesh.

Les imprimantes 3Ds à notre portée ont des tailles d'impressions assez variables de 10 à 30 centimètres. Une pièce à imprimer en 30 par 30 centimètres avec une imprimante possédant un plateau de 15 par 15 centimètres est découpé en 4 pièces à assembler entre elles par des attaches. Les anciens développeurs du projet ont donc opté pour une génération de maillages sous forme de pièces de puzzle assemblables permettant de produire des cartes de surfaces plus importantes à celles imprimables sans trop de complexité pour l'utilisateur.

L'image est chargée en mémoire sous forme de matrice. Plusieurs boucles parcourent cette matrice pour que les pixels en niveau de gris composant l'image soient transformés en sommets. Ceux-ci forment une couche en surface. D'autres boucles permettant ensuite de créer le socle de base à la carte et permettent de fermer le maillage.

Le choix d'exportation des maillages 3D s'est porté sur l'écriture dans un fichier OBJ (.obj). Ce format contient la description d'une géométrie 3D. Il suffit ensuite de donner de fichier à une imprimante. La figure suivante présente le résultat final de leurs travaux :

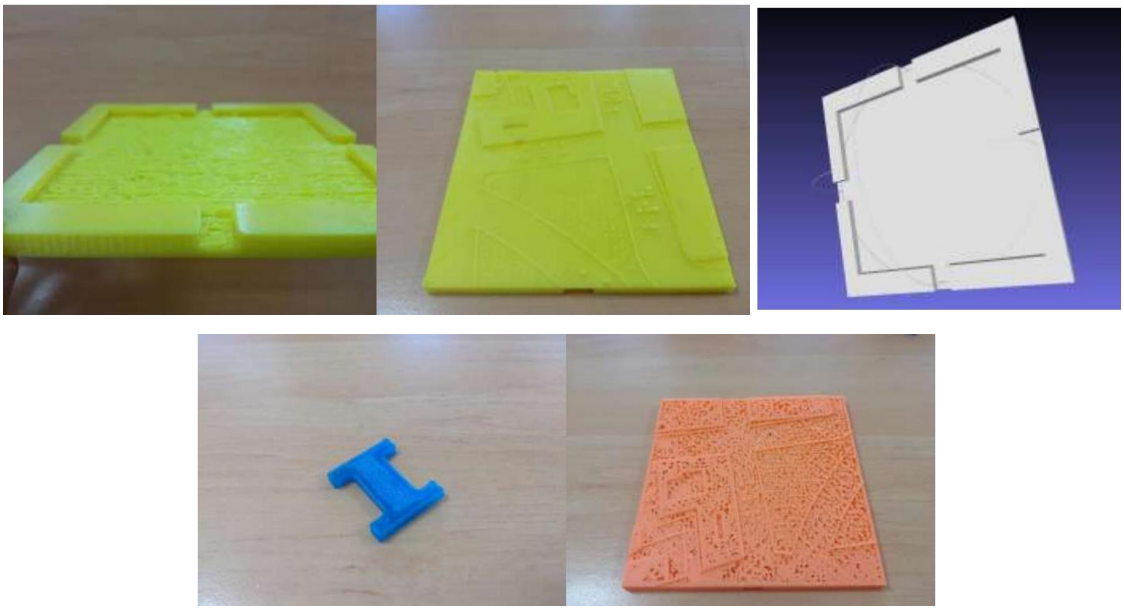


Figure 4 : Photos des tests d'impressions réalisés par les anciens étudiant

D) Contraintes et objectifs

Ce stage d'une durée de 8 semaines reprend le code existant des anciens étudiants. Dans un premier temps, le travail consiste à consolider ce code permettant de fabriquer un maillage 3D prêt à l'impression à partir d'une carte en niveau de gris. Dans un second temps, il est également question d'identifier et implémenter des modifications du maillage dans le but de faciliter son impression et d'améliorer l'ergonomie de l'objet 3D résultant pour des utilisateurs non-voyants.

Factorisation

Dans cette partie, il s'agit de reprendre le code existant. Celui-ci est implémenté assez simplement en français. Un travail de renommage, de traduction, de simplification et de factorisation permettrait de partir sur des bases propres, robustes et maintenables. Cette partie me permettra de me familiariser rapidement avec le code. Les algorithmes sont assez gourmands en ressources, il est donc également question de libérer un peu d'espace mémoire pour rendre le traitement plus performant.

Simplification de l'objet 3D

L'objet 3D résultant est composé uniquement de trimesh. Un des objectifs de ce stage est d'améliorer et de simplifier au maximum ces objets. Plusieurs pistes sont envisagées telles qu'un changement de structure de données*, d'utilisation de librairies* externes, de simplification et de décimation* ou d'utilisation d'un autre type de géométrie pour le maillage.

Améliorations

Une des améliorations envisagées par les anciens étudiants était de trouver une solution pour permettre d'écrire un identifiant au dos de la carte générée. Le socle a été creusé afin d'écrire au verso les numéros des différentes parties. Cependant après impression, ils se sont rendu compte que le dessous du socle était trop rugueux et par conséquent il est impossible d'imprimer un rendu propre. Des pistes ont néanmoins été explorées. Il s'agit donc de réfléchir en s'appuyant sur ces pistes et des normes existantes sur l'implémentation de cette amélioration.

II) ANALYSE ET CONCEPTION

A) Factorisation et amélioration du code

Utilisation de GitHub

Les sources du projet se trouvaient sur la forge de l'Université d'Auvergne. Nous avions décidé de migrer sur GitHub* pour poursuivre le projet. Un premier travail a donc été de cloner le projet de la forge sur GitHub. De plus, un fichier README au format Mardown* a été créé pour décrire le projet, sa structure, son installation et sa compilation.

Nettoyage et factorisation

Une factorisation du code a été nécessaire tout au long du projet. J'ai commencé par supprimer les méthodes et les variables inutiles puis documenter et traduire le code en anglais. J'ai ensuite pu factoriser certaines méthodes pour optimiser au maximum le code. Beaucoup de méthodes et d'attributs étaient statiques. Ceux-ci ont la particularité de pouvoir être utilisés sans instance d'objet. J'ai réalisé une légère refonte de la structure des classes afin d'obtenir une implémentation beaucoup plus orientée objet pour permettre une meilleure évolution de la structure. En annexe 1 et 2, se trouvent des diagrammes de classes de l'application avant et après changement.

Interface graphique

Pour simplifier l'interface, j'ai intégré la fenêtre de paramétrage directement dans la fenêtre principale dans le but de pouvoir modifier ces paramètres plus aisément pour recommencer si les objets générés ne nous conviennent pas. J'ai également trouvé utile d'ajouter une partie permettant d'afficher les objets générés en 3D. Dans la version actuelle de l'application, cette fonctionnalité ne fonctionne pas puisque la structure de données a été changée à l'aide d'une librairie par la suite. Cette librairie est détaillée dans la partie suivante. En revanche, j'ai ajouté une action dans le menu qui permet d'importer un fichier au format .obj dans cette vue pour pouvoir visionner un objet. Une liste est également présente sur la gauche de la fenêtre principale qui liste tous les objets de type maillage généré. Voici une capture d'écran de la nouvelle fenêtre :

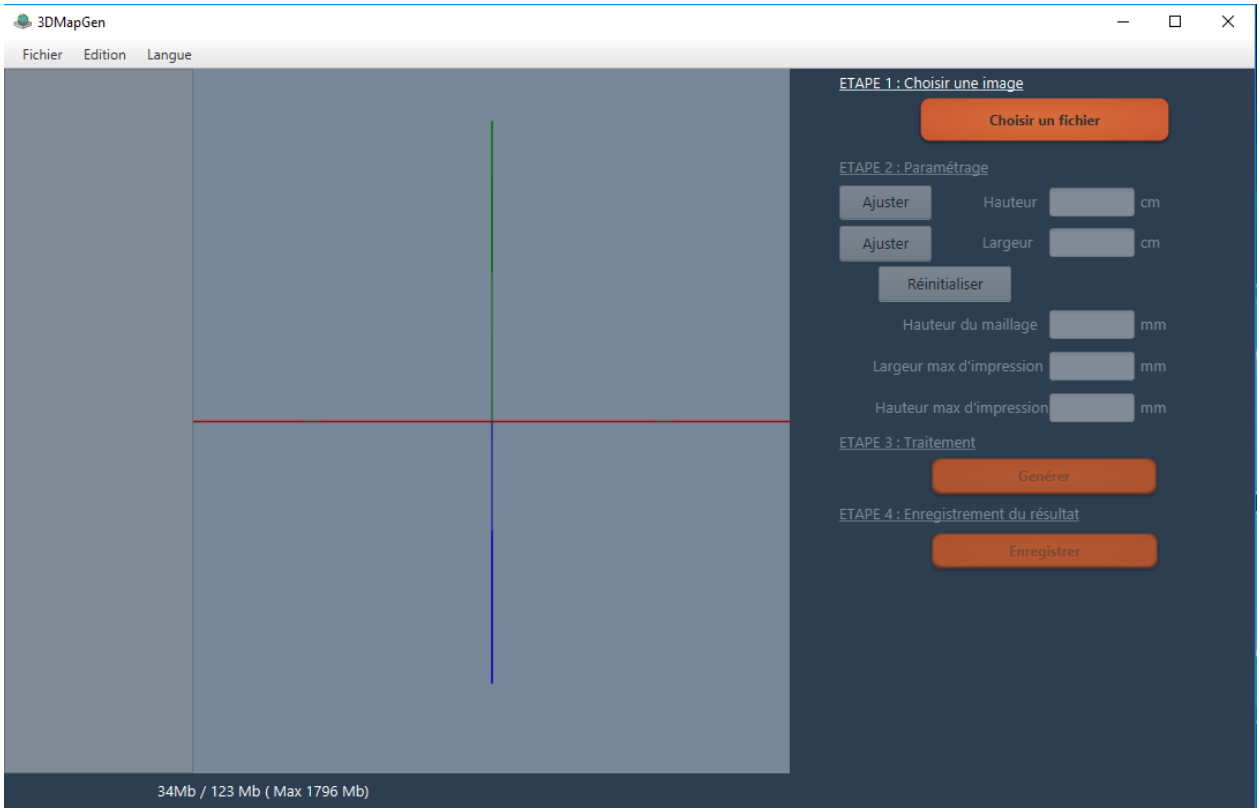


Figure 5 : Capture d'écran de l'interface graphique de l'application

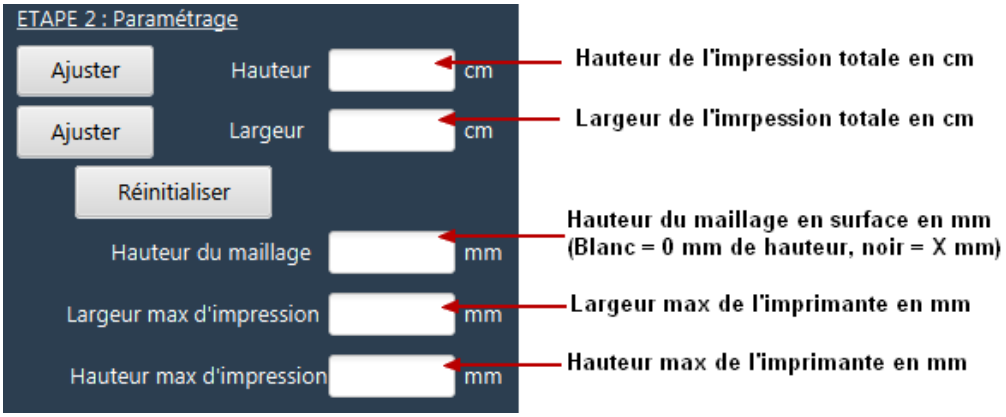


Figure 6 : Capture d'écran zoomée et légendée de la partie paramétrage

Configuration

Beaucoup de variables correspondaient à des valeurs constantes et étaient éparpillées dans le code. Pour rendre la modification de ces constantes plus facile et disponible sans avoir à lire tout le code, j'ai ajouté un fichier de configuration permettant d'indiquer ces valeurs tel que certaines dimensions fixes de la carte, les dimensions des clips ou les noms des fichiers et dossiers qui seront exportés.

Style

Le système de changement de style d'affichage était assez lourd puisque chaque item de l'affichage était changé un par un lorsque l'utilisateur choisit de changer de thème. Il fallait donc rajouter le traitement pour chaque item dans l'affichage pour tous les styles différents. J'ai remplacé cette fonctionnalité par une méthode qui change la feuille de style utilisé par les items Java FX. Pour ajouter un nouveau style, il suffit maintenant d'ajouter un fichier "themeX.css" dans le dossier stylesheet possédant les mêmes noms de classes que les autres fichiers. Il suffit ensuite d'ajouter une méthode de chargement de ce fichier sur le même modèle que les autres.

Internationalisation

J'ai profité de ces changements pour implémenter un système de changement de langue dynamique dans l'application. Grâce à une classe Java nommée Ressource Bundle, j'ai pu charger et exploiter des fichiers de ressources locales spécifiques. Le français et l'anglais ont été ajouté.

Pour ajouter d'autres langues, il faut ajouter un fichier de propriété (.properties) de la forme lang_"CODEALPHA2". Les codes alpha2 sont une représentation par une abréviation du pays et de la langue. Il faut également ajouter un nouvel item dans le menu langage, créer une méthode d'action sur le bouton créé puis initialiser le texte dans la fonction de rechargement de la même façon que les autres langues présentes.

Indication mémoire

Il était également intéressant de connaître la consommation mémoire pendant la génération des cartes. En bas de la fenêtre principale, j'ai ajouté un indicateur d'utilisation de la mémoire système. Un sous-processus permet d'actualiser ces indications toutes les secondes mais Java FX empêche l'affichage d'être mis à jour pendant la génération d'un maillage. L'indication en temps réelle est donc visible pour le moment uniquement en mode console lorsqu'un gros traitement a lieu.

B) Changement de la structure du maillage

Les précédents étudiants ont réalisé leur propre structure de données pour représenter un maillage. Il était intéressant de chercher des projets ou librairies existantes dans le but d'optimiser la structure du code et la génération de maillage et pouvoir utiliser des fonctionnalités de modification de celui-ci.

J'ai donc effectué des recherches sur les librairies existantes permettant de simplifier la structure actuelle et qui pourraient modifier un maillage par décimation ou simplification. La majorité des librairies sont développées en C++. Le développement de librairies en Java est en cours, c'est pourquoi il n'y a pas beaucoup de projets mis en place pour le moment. Voici un tableau récapitulatif des librairies étudiées :

HE_MESH (Java)	Librairie Java permettant de créer et manipuler un maillage de type polygone. Elle ne permet pas l'affichage puisqu'elle est destinée à être utilisée avec une autre librairie appelé Processing. Le projet est public et maintenu mais la documentation est assez incomplète.
toxiclibs (Java)	Classe permettant de construire, manipuler et exporter des maillages triangulaires. Le maillage est construit face par face. La classe réutilise automatiquement les sommets existants et peut créer des normales de sommet lisses. Les sommets et les faces sont facilement et rapidement accessibles. Avantage : possède des méthodes d'export d'un objet au format OBJ et STL et ne duplique pas les sommets. Cependant, elle ne possède pas de méthodes de simplification et de décimation de maillage.
MeshDecimation (Java)	Projet complet possédant des classes de structures de maillages et possédant des opérations sur celles-ci tel que la décimation ou la simplification.
CGAL (C++)	Bibliothèque de calcul géométrique écrite en C++. Elle permet la génération de maillages 2D et 3D et du traitement sur le maillage. CGAL étant écrit en c++, il n'est pas possible de l'utiliser avec l'application java directement mais en utilisant un système de binding de code.
LWJGL (Java)	Librairie Java qui permet l'accès multiplateforme aux API natives utiles dans le développement d'applications graphiques (OpenGL). LWJGL est open source et est utilisée pour le développement de jeux.
OpenMesh (C++)	Structure de données générique et efficace pour représenter et manipuler des maillages polygonaux.

Figure 7 : Tableau comparatif des différentes librairies étudiées

Nous avons décidé d'opter pour l'utilisation de la librairie HE_MESH. C'est une puissante bibliothèque pour créer une géométrie et constitue un excellent outil pour explorer les possibilités d'impression. La logique de base dans la création de formes avec HE_MESH repose sur une compréhension des fondamentaux de la géométrie 3D et plus précisément sur la géométrie des polygones. La librairie possède une structure et un ensemble de méthodes qui permettent la simplification de maillage.

Les objets créés avec des maillages polygonaux stockent différents types de données: les sommets, les arêtes, les faces, les polygones et les surfaces. Ces données contiennent des informations sur la façon dont chacune d'entre elles sont interconnectées et donne ainsi la possibilité d'y accéder. Il existe quatre classes de base principales qui composent HE_MESH, nécessaires pour créer nos objets graphiques :

- Les classes préfixées par HE_ spécifient des classes de base pour accéder à des données spécifiques concernant un maillage.
- Les classes préfixées par HEC_ spécifient des créateurs pour créer et configurer les formes géométriques de base.
- Les classes préfixées par HEM_ précisent les modificateurs pour appliquer un certain nombre de modifications aux formes.
- Les classes préfixées par WB spécifient une multitude d'outils mathématiques et de fonctionnalités de rendu.

Toutes les formes sont donc créées à l'aide de la classe HE_Mesh et des classes HEC_. Les classes HEM_ sont utilisées pour modifier et interagir avec les formes. Cela permet d'avoir un espace de travail beaucoup plus facile à gérer car la classe HE_Mesh est dédiée à donner accès aux éléments de données de base d'un maillage alors que la classe HEC_ est une famille de classes qui nous permet de choisir différentes formes.

Cependant, nous nous sommes aperçu que la génération était beaucoup plus longue en utilisant les fonctionnalités de simplification car le maillage est trop dense. Une simplification du maillage de l'objet est donc indispensable avant de créer l'objet final en utilisant la structure HE_Mesh. La partie suivante détail les transformations réalisées sur le maillage et l'algorithme de génération pour le rendre plus simple.

C) Simplification du maillage

Changement de type de maillage

Les faces étaient composées de quatre points qui formaient deux faces triangulaires. Dans un premier temps, j'ai effectué la transformation de ces faces à des faces de type polygonale. La majeure partie des faces sont donc maintenant représentées par des quadrilatères mais, certaines faces latérales peuvent former des faces polygonales qui possèdent beaucoup plus de points. Ce changement réduit considérablement le nombre de faces. Cela permet de rendre l'export au format OBJ beaucoup moins volumineux et la génération du maillage plus rapide. Le schéma suivant représente l'évolution du maillage :

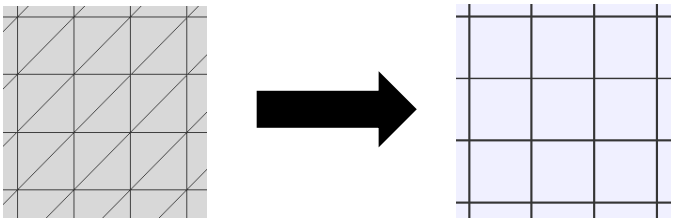
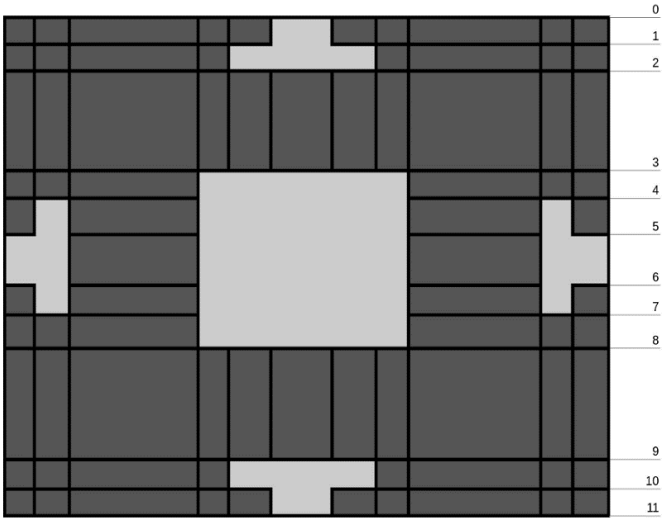


Figure 8 : Schéma représentant l'évolution du maillage de l'objet

Simplification du socle de base

Le socle de base possède un maillage beaucoup trop dense et peut être réduit. Une simplification peut être faite car les attaches feront toutes la même taille et un identifiant de carte (lettre + chiffre) sera dans une surface creusée dont la taille ne bougera pas. L'idée est d'obtenir un maillage du socle composé de 12 x 12 (144) points.



0	0
1	Hauteur Interne Clip / 2
2	Hauteur Totale Clip / 2
3	(Largeur Map – Taille Carré Lettre) / 2
4	(Largeur Map – Largeur Extérieur Clip) / 2
5	(Largeur Map – Largeur Intérieur Clip) / 2
6	(Largeur Map + Largeur Intérieur Clip) / 2
7	(Largeur Map + Largeur Extérieur Clip) / 2
8	(Largeur Map + Taille Carré Lettre) / 2
9	(Largeur Map – Hauteur Totale Clip)
10	(Largeur Map – (Hauteur Interieur Clip /2))
11	Largeur Map

Figure 9 : Schéma représentant la structure du maillage sous l'objet

Nouvelle méthode de génération

L'algorithme commence par créer et référencer tous les points de la carte en surface dans un objet Java de type TreeMap* en parcourant une image ou une partie d'image. Il en profite pour stocker en plus, tous les points des bordures dans un autre. Il poursuit avec l'indexation de tous les points de la base de la carte. Les coordonnées de ces points sont variables grâce à la simplification du socle de base mais, possèdent la même structure quel que soit le maillage généré. Deux TreeMaps sont donc construites de façon à avoir une partie haute et basse du socle.

Des polygones composés de quatre points sont créés pour construire les faces en surfaces à partir des points référencés. Les faces latérales sont ensuite construites et découpés en plusieurs parties : une première face composée de tous les points latéraux de la surface de la carte et de ceux de la partie juste inférieure pour conserver la densité de points. Nous construisons ensuite d'autres faces qui permettent la liaison des faces sur la droite et la gauche de l'espace du clip et de la face créée précédemment. Enfin, les faces à droite et à gauche des polygones sont créées.

L'algorithme créé ensuite les faces sous la carte, puis construit un objet HE_Mesh à l'aide d'un constructeur de type HEC_FromPolygons à partir de la liste composée de face de type WB_Polygon. L'image suivante représente une capture d'écran du rendu :

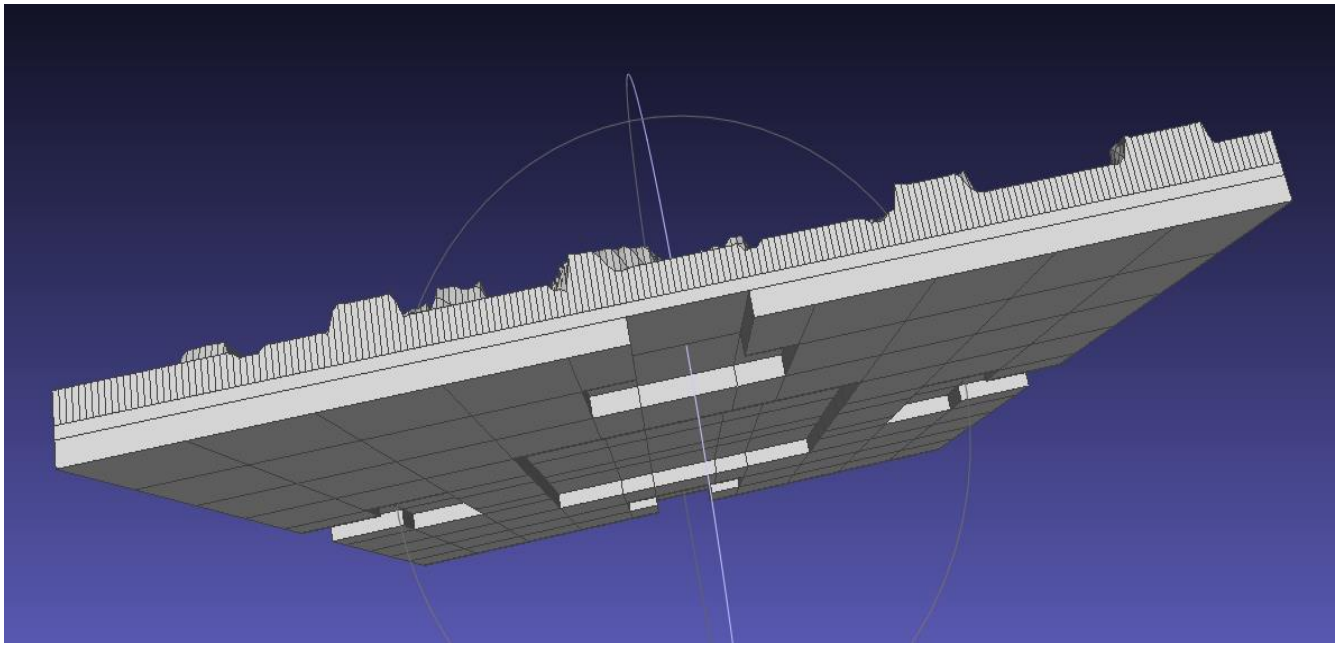


Figure 10 : Capture d'écran du rendu sous l'objet généré avec le nouvel algorithme

Les images suivantes représentent l'évolution du maillage en surface avant puis après le changement d'algorithme. À première vue, la transformation est subtile puisque le maillage est très dense mais, il est considérablement simplifié sans perte de qualité.

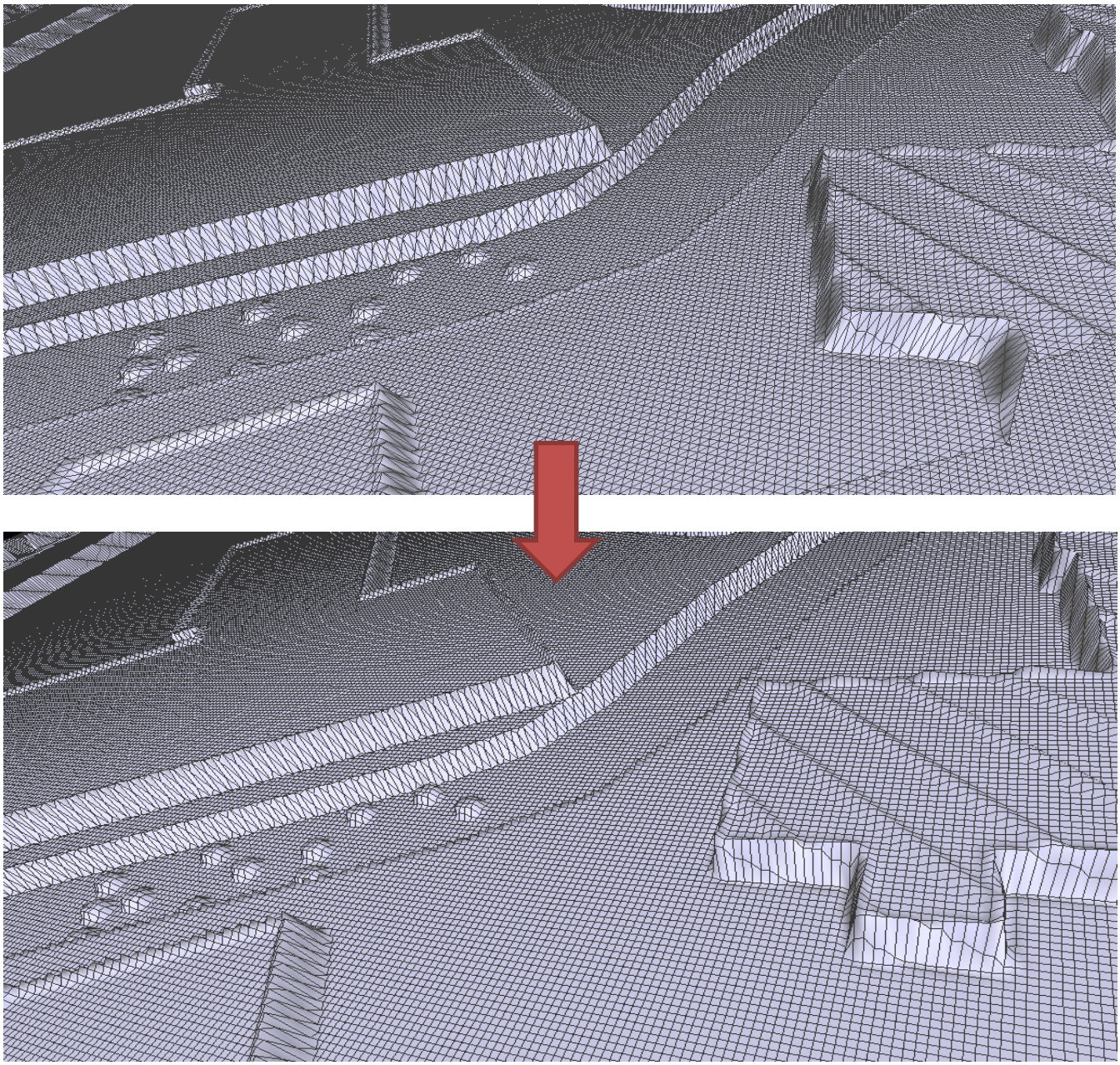


Figure 11 : Captures d'écran montrant l'évolution du maillage en surface avant puis après le changement d'algorithme

Amélioration de l'exportation

La librairie HE_Mesh propose des méthodes permettant d'exporter les maillages en fichier .obj avec ou sans les normales*. Les méthodes génèrent également des fichiers MTL qui ne nous sont pas utiles. Comme ces méthodes d'export sont statiques, il fallait recoder la fonctionnalité en se basant sur le code de la librairie.

D) Tests de génération

Comparaison de version

J'ai réalisé une série de tests permettant d'étudier le comportement des performances en matière de mémoire utilisée et de temps de traitement de l'application en fonction des différentes modifications apportées. Les données suivantes sont les résultats de cette étude réalisée avec un ordinateur disposant de 16 giga-octets de mémoire vive.

Etude réalisée avec la configuration suivante :
Intel(R) Core(TM) i7-3770S CPU @ 3,10GHz, 3101 MHz, 4 coeur(s), 8 Processeur(s) logiques(s)

A	Projet initial au début du stage
B	Changement de la structure de données du maillage avec HE_Mesh
C	Changement des TriangleMesh par des QuadMesh
D	Changement des Quads par des polygones, base simplifiée, export amélioré

Cas 1 : Image de 500 par 500 pixels à imprimer en 20 par 20 cm à l'aide d'une imprimante 10 par 10 cm (4 parcelles)

	A	B	C	D
Durée du traitement (secondes)	2	52	5	6
Poids des fichiers générés (Mo)	34,2	66,1	27,3	18,3
Consommation mémoire RAM (Mo)	<200	3000	1000	1000

Cas 2 : Image 1500 par 1500 pixels à imprimer en 60 par 60 cm à l'aide d'une imprimante 15 par 15 cm (16 parcelles)

	A	B	C	D
Durée du traitement (secondes)	12	Inconnue	> 120	48
Poids des fichiers générés (Mo)	323	Inconnue	Inconnue	169
Consommation mémoire RAM (Mo)	1500	OutOfMemoryError	OutOfMemoryError	2200

Cas 3 : Image 500 par 1000 pixels à imprimer en 15 par 30 cm à l'aide d'une imprimante 15 par 15 cm (2 parcelles)

	A	B	C	D
Durée du traitement (secondes)	3	> 300	35	17
Poids des fichiers générés (Mo)	70,2	Inconnue	54	38,2
Consommation mémoire RAM (Mo)	<300	Inconnue	2,5	1000

Figure 12 : Schéma représentant l'étude des performances de l'application

Beaucoup de ressources sont nécessaires pour générer une carte en 3D à partir d'une image. Plus l'image d'entrée possède une grande résolution, plus la génération sera longue et gourmande en mémoire sauf si elle est découpée en plusieurs parcelles. Parfois, la génération bloque (Out Of Memory). Nous pouvons constater que le projet actuel consomme plus de mémoire vive et la génération est plus longue. En revanche, le poids des fichiers générés a environ été divisé par deux. Il faut également prendre en compte le fait que nous obtenons un résultat avec un maillage plus simplifié qu'il n'était au début.

Tests d'impressions

En début de stage, une carte d'essai a été imprimé par l'atelier de l'ISIMA et nous a permis d'avancer avec un rendu réel. Cela nous a été utile pour connaître les premières modifications à prévoir tel que la réduction du poids du fichier à donner à l'imprimante. Voici des photos de l'imprimante 3D utilisée pour cette première impression. Sur la photo de gauche, on distingue le début d'impression de la base d'une carte.

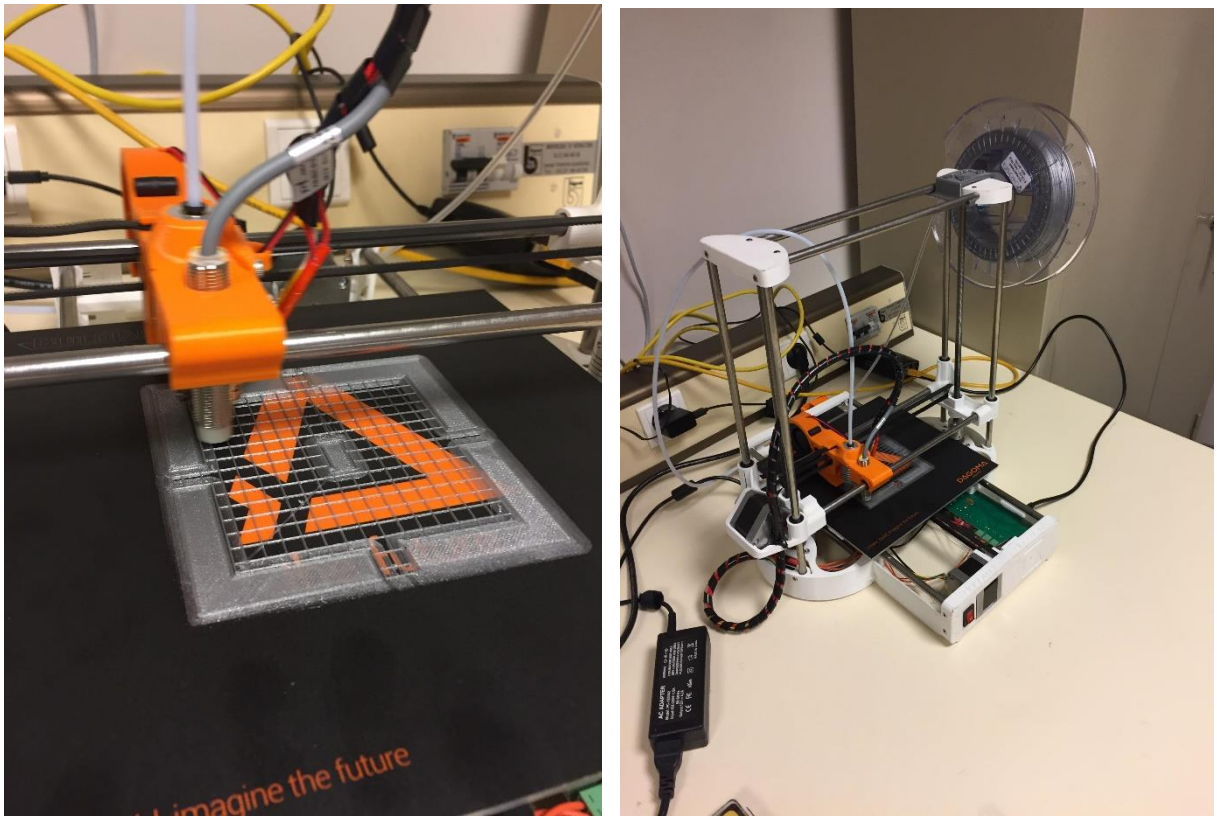


Figure 13 : Photos de l'imprimante 3D de l'atelier de l'ISIMA

Nous avons également été en relation avec le laboratoire SIGMake situé dans les locaux de l'école SIGMA à Clermont-Ferrand. Nous avons apporté des tests de cartes et de clips à imprimer. Cependant, l'impression n'a pas pu être réalisée. La plus grosse problématique de l'impression concerne les défauts de lissage du maillage. L'impression est plus complexe et difficile car le non-lissage de certaines parties du maillage entraine de trop forte vibration la tête d'impression et le rendu ne sera pas propre. De plus, cela pourrait abimer l'imprimante 3D. Il faudrait donc obtenir un maillage lissé dans les endroits où les surfaces sont crochues. Les textures générées sur les bâtiments quant à elles sont assez propres.

L'image suivante présente une capture d'écran du logiciel Ultimaker2, utilisé par SIGMake pour lancer une impression 3D. Les parties rouges représentent les parties les moins lisses à optimiser pour permettre une meilleure impression.

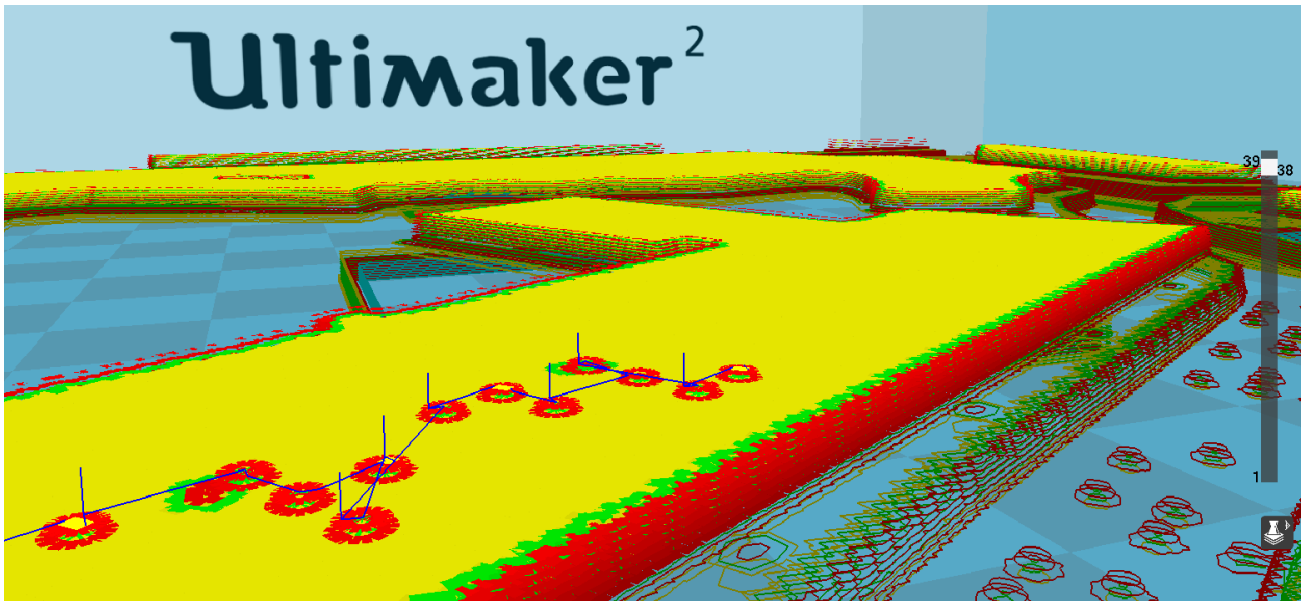


Figure 14 : Capture d'écran du logiciel Ultimaker2 d'un test d'impression d'une carte.

En revanche, dans la version actuellement désignée, les clips ont une épaisseur uniforme de 4 mm et ont une forme simple. Il nous a été proposé d'utiliser un processus de fabrication par découpage laser de plexiglas, qui est beaucoup plus rapide (ce qui peut être fait à partir du fichier 3D généré).

III) BILAN DU TRAVAIL

A) Résultats obtenus

Résultat du travail :

À ce stade, l'application est fonctionnelle et utilisable par une personne concernée par le projet. Un exécutable de l'application est disponible sur le GitHub du projet sous forme de fichier JAR*. Une documentation est également disponible sous forme de fichier README pour décrire les étapes d'installation de tous les éléments nécessaires au bon fonctionnement du projet. Des pages wiki sont présentes et décrivent la totalité de mon travail pour permettre aux futures personnes qui travailleront sur l'amélioration de l'application à pouvoir comprendre mes choix de développement.

L'accent a été mis sur l'ensemble des erreurs qui pourraient survenir lors de l'utilisation de l'application. L'ensemble du code a été commenté. Ainsi, la mise en place de telles améliorations apportera un plus à l'ensemble du projet.

Bilan temporel :

Au début du projet, j'ai énuméré un ensemble de tâches à réaliser en me basant sur les idées d'améliorations du rapport de stage des anciens étudiants. Très vite, cette liste de tâche s'est agrandie grâce aux nombreux échanges avec les intervenants de l'association Braille & Culture, les volontaires des ateliers d'impression de l'ISIMA et de SIGMA et des réunions hebdomadaires de projet. Tous les points n'ont pas pu être traités pendant ces huit semaines de stage. La prise en main du travail existant et la recherche d'une meilleure architecture ont pris la plus grande partie de mon temps.

B) Difficultés rencontrées

J'ai rencontré des difficultés afin de mener à bien le projet qui m'a été confié. En effet, j'ai mis un certain temps à prendre en main et comprendre toutes les subtilités de l'application existante. J'ai pu me baser sur le rapport de stage des anciens étudiants mais l'étude et la prise en main du travail existant ont donc pris une grande partie de mon temps.

Il était également nécessaire d'imaginer la solution la plus intéressante en fonction de l'attente afin d'améliorer et simplifier la génération au maximum. Il fallait ainsi envisager une disposition claire et adaptée pour permettre d'avancer.

En ce qui concerne le code et la prise en main de la librairie HE_Mesh, cela n'a pas été rapide, il n'a pas été facile de gérer certaines erreurs et de m'approprier le code existant. Une bonne documentation accompagne cet outil de développement. Elle était parfois incomplète, mais elle m'a permis d'obtenir les noms de fonctions qui m'étaient indispensables et connaître l'architecture de la librairie.

Les objets 3Ds de tests générés étaient parfois complexes à étudier. Plusieurs logiciels tels que MeshlLab ou Blender3D ont pu être utilisés pour les visualiser. Parfois les fichiers objets étaient beaucoup trop volumineux pour être ouvert avec de tels logiciel, cela à poser quelques difficultés. Des problèmes ont également concerné l'orientation des faces et des arêtes. En effet, en visualisant le maillage, j'obtenais parfois des zones sombres correspondant en fait à des faces mal orientées. Dans certains cas, il a été difficile de bien les réorienter.

L'exécution du traitement de la génération de maillages a aussi été un problème puisqu'elle pouvait prendre énormément de ressources. Parfois le traitement pouvait s'arrêter brutalement pendant un test. Ainsi, il fallait optimiser l'algorithme pour le rendre moins gourmand en mémoire et en utilisation du processeur au fil des changements apportés.

C) Améliorations et prolongements

Si un délai supplémentaire me serait accordé pour continuer le travail qui m'a été confié, j'aurai plusieurs améliorations à apporter à des parties de l'application présentée dans ce rapport que je vais vous énoncer par ordre de priorité.

Lissage du maillage

Tout d'abord, il est indispensable de réfléchir à une solution permettant de corriger le problème d'un maillage non lissé pour envoyer la pièce à l'impression tel que décrite dans la partie D. Sans celle-ci, il sera très difficile d'imprimer les cartes. Cette modification pourrait être apporté en amont du traitement sur l'image que l'on fournit à l'application.

Identifiant de carte

Une idée d'amélioration concerne également l'ajout d'identifiant sous la carte. Actuellement le dessous de celle-ci est creusé par un espace carré de dimensions égale sous toutes les cartes générées pour pouvoir insérer des lettres et des chiffres en 3D plaqué contre une face. Cette partie n'a pas été traitée, en revanche, après de brèves recherches, j'ai trouvé qu'il était possible tout d'abord de générer une lettre 3D avec une chaîne de caractères et à l'aide d'une librairie. Puis dans un second temps, de la combiner aux maillages 3Ds de la carte en utilisant un objet de type HE_Mesh.

Viewer 3D

Au début du projet, j'ai trouvé utile d'ajouter un espace dans la fenêtre principale de l'application permettant de voir l'objet en 3D avant de l'exporter. Malheureusement, lorsque j'ai changé le changement de structure du maillage cette fonctionnalité ne pouvait plus être utilisée à cause de l'incompatibilité entre la structure 3D d'un objet Mesh Java FX et les objets HE_Mesh. Cette fonctionnalité pourrait être reprise en essayant d'implémenter un rendu 3D en utilisant la librairie Processing et un Applet Java*.

Regrouper l'export des clips

Une autre piste pourrait être étudiée. Il s'agit de regrouper le plus de clip nécessaire à l'assemblage des cartes entre elles dans le même fichier suivant la place restante. Ainsi, plusieurs clips pourraient être imprimés en même temps et économiser des impressions.

BILAN TECHNIQUE

Ce stage avait pour objet d'améliorer une application permettant de générer des cartes tactiles. Ainsi, j'ai mis en place des fonctionnalités et des améliorations rendant l'application plus robuste et une génération plus simplifiée capable d'être utilisé par le plus grand nombre. De plus, mon travail pourra facilement être repris par les futures personnes qui travailleront sur ce projet grâce une sauvegarde de toutes mes recherches et ma documentation disponible sous forme de page wiki.

Ces huit semaines de stage m'ont permis d'appliquer concrètement les connaissances issues de mes formations antécédentes et de compléter ces acquis en matière de programmation en Java et Java FX, mais également avec ma formation actuelle, notamment en matière d'infographie, de modélisation et de création 3D.

Grâce à de nombreuses réunions, j'ai pu découvrir plus en profondeur le monde de la malvoyance qui m'était inconnu jusqu'à présent. J'ai pu prendre connaissance des outils et des méthodes utilisées par des malvoyants pour se déplacer ou se repérer mais aussi de la lecture et de l'écriture braille. J'ai également pris conscience que l'aboutissement de ce projet pourrait être un réel plus en matière de déplacement au quotidien pour ces personnes.

Ce qui a finalement pris le plus de temps dans le projet a été de réfléchir à la meilleure architecture des éléments nécessaires au développement des fonctionnalités. En effet, il existe plusieurs architectures possibles pour bien organiser un projet, mais il a été nécessaire de créer une structure pour permettre une meilleure évolution de l'application. Ces améliorations permettront à l'équipe travaillant sur ce projet de repartir avec une application plus solide.

CONCLUSION

Dans le cadre de ce projet, les objectifs ont été très clairs. J'avais pour mission de consolider et d'ajouter des fonctionnalités à un logiciel de génération de carte 3D en Java. Afin de réaliser le travail, il a été possible d'utiliser la librairie HE_Mesh. Cela m'a permis d'améliorer le développement de l'application grâce aux outils que cette librairie apporte. L'analyse de la conception avant le développement et les réunions de projet hebdomadaires ont été des étapes clefs pour savoir dans quelle direction s'orienter. J'ai ainsi décrit dans ce rapport l'analyse et le développement de l'application avec de nombreuses illustrations et explications d'utilisations.

Ce projet était une opportunité qui s'est révélée très intéressante et enrichissante dans la mesure où il a consisté en la découverte du monde de la malvoyance. Il m'a permis de me confronter à un nouveau mode de développement, avec des contraintes matériels, logiciels et de programmation. J'ai donc dû réagir en conséquence et de manière autonome afin de les régler. Ce projet a également permis d'approfondir certaines méthodes de travail, à savoir la plate-forme de collaboration GIT, qui s'est avérée très utile pour le travail en équipe. De plus, travailler de manière autonome dans une équipe de trois autres étudiants était une excellente expérience qui m'a permis d'avoir un avant-goût du travail d'équipe professionnel en entreprise ou en laboratoire, en plus de m'avoir permis de développer mes compétences humaines, tel que la communication, l'ouverture d'esprit et l'organisation. J'ai ainsi pu appliquer les connaissances et les méthodes de travail préalablement étudié en cours à la faculté de médecine de Clermont-Ferrand spécialité Technologies Biomédicales.

ENGLISH SUMMARY

I am a student in a four-year degree in Biomedical Technologies Sciences at the Clermont-Ferrand Faculty of Medicine. It is an eight-week internship carried out at L.I.M.O.S laboratory as part of the university curriculum. I have the opportunity to work on a project, which enabled to help a starting search project to help persons with impaired vision. The objective of this internship was to improve a Java application to generate a 3D object with an image in level of grey.

Mr. Jean-Marie FAVREAU, lecturer at the University of Clermont-Ferrand and specialist in algorithmic geometry, put this topic forward. This internship takes place in a context of a partnership between a research team of the National Institute of Geographic and Forestry (IGN) and the G4 theme of LIMOS.

I started by adding functionality and improving the existing application. This step allowed me to take charge of the project. Then I improved and simplified the mesh by using some libraries with several tests and many weekly meetings.

Finally, after many weeks, I have implemented features and improvements making the application more solid and a treatment process allows to be used by as many people as possible. In addition, future people who will work on this project can easily pick up my work. It was possible to use the library HE_Mesh. This allowed me to improve the development of the application thanks to the tools that this library brings.

Overall, this project was a good learning experience. It enabled me to develop my personal skills such as the capacity to working autonomously and improve my organizational competence, as well as teamwork skills, which was a very positive experience and a useful way to combine my different abilities. I also improved my knowledge in the field of computer graphics, modeling and 3D creation. I gained experience in programming skills, especially in Java and JavaFX Programming.

BIBLIOGRAPHIE

Sites web

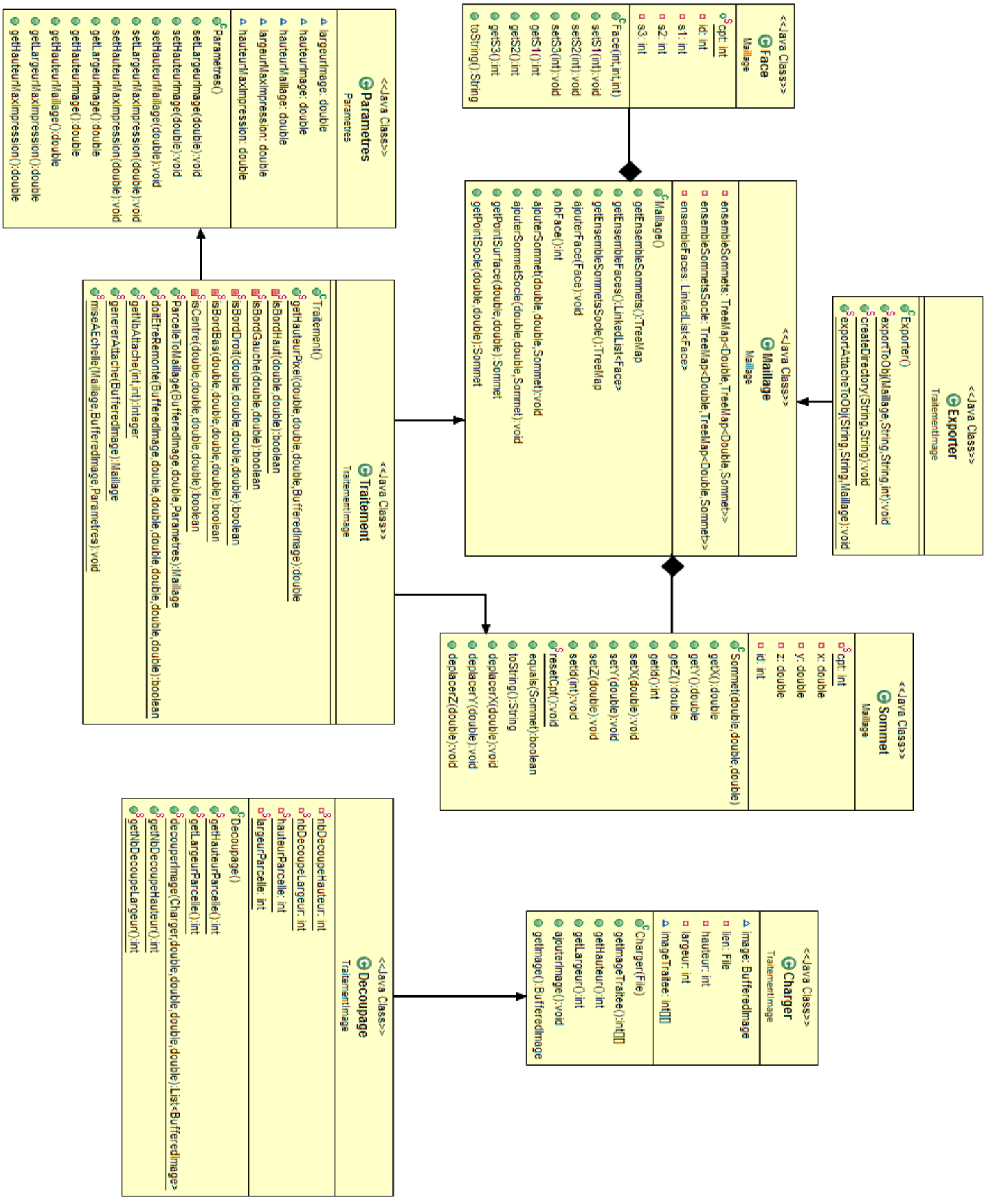
- https://github.com/PierreCharles/3DMapGen
- https://docs.oracle.com/javase/8/docs/api/
- www.dummies.com/programming/java/javafx-add-a-mesh-object-to-a-3d-world/
- http://www.interactivemesh.org/
- http://code.makery.ch/blog/javafx-dialogs-official/
- http://docs.oracle.com/javafx/8/3d_graphics/sampleapp.htm
- www.enseignement.polytechnique.fr/informatique/INF555/TD/Jcg/INF555-Jcg.html
- https://github.com/wblut/HE_Mesh
- http://www.flipcode.com/archives/The_Half-Edge_Data_Structure.shtml
- http://freeartbureau.org/fab_activity
- https://github.com/AmnonOwed/CAN_GenerativeTypography/blob/master/3D/Basic3DType/Basic3DType.pde
- http://www.ricardmarxer.com/geomerative/
- https://processing.org/
- http://jmonkeyengine.org/
- https://www.lwjgl.org/
- https://www.openmesh.org/
- http://www.cgal.org/
- http://www.objectaid.com

LEXIQUE

- **Applet :** Classe permettant de fournir des fonctionnalités interactives.
- **Décimation :** Type de simplification de maillage.
- **GitHub :** Site offrant l'hébergement et proposant la possibilité de collaborer et de contribuer à des projets.
- **Image en niveaux de gris** : Image dont les couleurs vont varier du blanc au noir.
- **Impression 3D :** Procédé de fabrication de pièces en volume par ajout ou agglomération de matière.
- **JAR :** Format de fichier utilisé pour distribuer un ensemble de classes Java constituant l'ensemble d'un programme.
- **Java FX :** Bibliothèque de création d'interface graphique multiplateforme officielle du langage Java.
- **Librairie** : Ensemble normalisé de classes, de méthodes ou de fonctions permettant d'offrir des services à d'autres logiciels.
- **Maillage :** ensemble de formes géométriques disposées de manière à modéliser des objets. Il est constitué de sommets, connectés les uns aux autres par des faces.
- **Mardown :** Langage de balisage léger offrant une syntaxe facile à lire et à écrire.
- **Normale :** Vecteur directeur.
- **Pixel :** Unité de base permettant de mesurer la définition d'une image numérique matricielle.
- **Structure de données :** Manière abstraite d'organiser les données pour les traiter plus facilement.
- **TreeMap :** Classe Conteneur Java permettant de stocker des couples (clé, valeur) dans une structure d'arbre binaire.
- **Variable :** Symbole qui renvoie à une position de mémoire contenant des valeurs.

Table des annexes

Annexe 1 : Diagramme de classes au début du stage



Annexe 2 : Diagramme de classes à la fin du stage