Stage ingénieur

Support de la modélisation solide dans l'impression 3D : Évaluation au travers du format AMF

Virgile Daugé

Année 2015–2016

Stage de 2 année réalisé dans l'entreprise Parallel Geometry

en vue de l'obtention du diplôme d'ingénieur de TELECOM Nancy

Maître de stage : Jean-François Rotgé

Encadrant universitaire : Thibault Cholez

Déclaration sur l'honneur de non-plagiat

Je soussigné(e),

Nom, prénom : Daugé, Virgile

Élève-ingénieur(e) régulièrement inscrit(e) en 2e année à TELECOM Nancy

Numéro de carte de l'étudiant(e) : 31110541

Année universitaire : 2015–2016

Auteur(e) du document, mémoire, rapport ou code informatique intitulé :

Support de la modélisation solide dans l'impression 3D :
Évaluation au travers du format AMF

Par la présente, je déclare m'être informé(e) sur les différentes formes de plagiat existantes et sur les techniques et normes de citation et référence.

Je déclare en outre que le travail rendu est un travail original, issu de ma réflexion personnelle, et qu'il a été rédigé entièrement par mes soins. J'affirme n'avoir ni contrefait, ni falsifié, ni copié tout ou partie de l'œuvre d'autrui, en particulier texte ou code informatique, dans le but de me l'accaparer.

Je certifie donc que toutes formulations, idées, recherches, raisonnements, analyses, programmes, schémas ou autre créations, figurant dans le document et empruntés à un tiers, sont clairement signalés comme tels, selon les usages en vigueur.

Je suis conscient(e) que le fait de ne pas citer une source ou de ne pas la citer clairement et complètement est constitutif de plagiat, que le plagiat est considéré comme une faute grave au sein de l'Université, et qu'en cas de manquement aux règles en la matière, j'encourrais des poursuites non seulement devant la commission de discipline de l'établissement mais également devant les tribunaux de la République Française.

Fait à Montréal, le 18 décembre 2016

Signature :

Stage ingénieur

Support de la modélisation solide dans l'impression 3D : Évaluation au travers du format AMF

Virgile Daugé

Année 2015–2016

Stage de 2e année réalisé dans l'entreprise Parallel Geometry

en vue de l'obtention du diplôme d'ingénieur de TELECOM Nancy

Virgile Daugé
1, rue du Général de Castelnau
54600, Villers-lès-Nancy
t+33 6 63 46 19 65
virgile.dauge@telecomnancy.net

TELECOM Nancy
193 avenue Paul Muller,
CS 90172, VILLERS-LÈS-NANCY
+33 (0)3 83 68 26 00
contact@telecomnancy.eu

Parallel Geometry
400, ave Atlantic Bureau 100
H2VlA5, Outremont, Canada
+1 (514) 277-6413

Maître de stage : Jean-François Rotgé

Encadrant universitaire : Thibault Cholez

Remerciements

J'adresse mes remerciements aux personnes qui m'ont aidé dans la réalisation de ce stage.

En premier lieu, je remercie Mr Denis Akzam, CEO de Parallel Gometry pour son accueil au sein de l'entreprise.

Je remercie également Jérémie Farret pour le temps qu'il m'a consacré, le suivi constant apporté ainsi que ses conseils avisés aidant au bon déroulement du stage.

Je tiens enfin à remercier Jean-François Rotgé pour son aide théorique, ainsi que pour de nombreux échanges enrichissants, qui m'ont permis d'appréhender des concepts nouveaux et prometteurs.

Table des matières

1 Introduction

La fabrication additive a connu une croissance exponentielle ces dernières années, autant par la multiplication des domaines d'activités que par l'explosion des capacités des imprimantes. De nombreux standards définissent des formats de fichiers aux utilisations diverses dans le monde de la fabrication additive. Néanmoins, une grande partie du secteur utilisait, et utilise toujours le format de fichier assez pauvre baptisé STL [1]. Il devenait alors évident qu'un nouveau format devait être standardisé afin de suivre au mieux les nécessités et les capacités grandissantes du domaine.

C'est dans ce contexte que l'organisation internationale de normalisation (ISO) ainsi que l'American Society for Testing and Materials (ASTM) ont collaboré pour créer la norme AMF pour additive manufacturing File Format, qui est aujourd'hui la norme internationale pour l'impression 3D. La norme AMF est un standard de fichiers couvrant les besoins actuels et futurs de la fabrication additive. Elle vient remplacer l'ancien mais toujours très utilisé fichier STL, en apportant une compression supérieure permettant de gagner sur le volume nécessaire au stockage. Ce gain de place a fait le succès de ce format de fichier, néanmoins ce n'est qu'une de ses caractéristiques prometteuses. Comme en STL, l'objet est représenté par sa surface, sous forme de meshs [2] composés d'assemblages de triangles. Autre nouveauté apportée par la norme AMF, la gestion de matériaux, et par extension, de couleurs. C'est là le principal atout de cette norme, quoique peu utilisé à l'heure actuelle, certainement par manque de connaissances et de documentations.

La mise à jour de la norme AMF n'était pas au goût du jour, jusqu'à ce qu'un nouveau consortium fasse sont entrée, 3MF, poussant un format éponyme, qui partage de nombreux points communs avec AMF, sur le devant de la scène. Il devint alors nécessaire pour les organismes gérant la norme AMF de la mettre à jour, pour conserver une certaine compétitivité et ainsi assurer la pérénité de la norme. Malheureusement, les initiateurs et principaux acteurs du projet, à savoir Jonathan Hiller et Hod Lipson, ne sont plus actifs à ce jour. Cela a pour conséquence une méconnaissance des possibilités d'AMF et de son logiciel de référence, AMFTools. Il s'avère donc nécessaire pour le groupe de travail de la norme de faire le point sur les capacités de la norme en l'état, avant de songer à y apporter des modifications.

C'est là qu'intervient ce stage, ayant pour but de cerner aux mieux les possibilités offertes actuellement par le Couple AMF AMFTools, en particulier les possibilités de représentations solides. En effet, l'équipe de Parallel Geometry, experte dans le domaine, y entrevoit une possibilité de démontrer les inconvénients de la représentation polygonale actuellement utilisée, et par extension les bénéfices de la représentation solide, ou mieux encore de l'arithmétique des formes.

L'objectif est donc dans un premier temps la réalisation d'une batterie de tests complets permettant d'illustrer les différentes fonctionnalités prévues par la norme, afin d'identifier les limites actuelles et proposer d'éventuels axes d'amélioration de la norme.

1. STL vient de Stéréolitographie.
2. un mesh, ou maillage, est une modélisation géométrique par des éléments bien définis, souvent des triangles.

2 Présentation de l'entreprise

Ce stage s'effectue à Montréal, au sein de l'entreprise P4BUS [1], filiale de Parallel Geometry [2] dédiée à l'exploitation des innovations et des brevets de Parallel Geometry, et plus particulièrement ceux s'appliquant à la fabrication additive. Ces innovations sont le fruit d'une recherche incessante menée par Jean-François Rotgé, dans la continuité de sa thèse sur l'arithmétique des formes, présentée dans la sous-section 3.1.1. Le groupe est en réalité une petite équipe aux compétences complémentaires. Bien qu'étant une petite entreprise comptabilisant moins de 10 employés, Parallel Geometry détient un savoir faire unique au monde, demandé par de nombreuses multinationales. Ils ont ainsi eu l'occasion de travailler en partenariat avec Intel ou encore l'agence Spaciale Canadienne sur de la simulation parallèle. En effet, bien que la 3D soit le coeur de métier de LLG, leur confortable avance dans de nombreux domaines de l'algorithmique leur permet d'apporter une réelle aide à ces grandes entités. Dans le cadre de la robotique cette fois, ils ont effectué en collaboration avec l'agence spaciale Canadienne des simulations de collision, ainsi que de dynamiques de contact sur bras robotisé équipant actuellement la station spaciale internationale. Des acteurs importants du domaine de l'informatique, comme Dassault Systèmes, font également appel à Parallel Geometry lorsqu'ils se retrouvent en difficulté face à un algorithme particulier. Effectuer mon stage au sein de cette entreprise représente donc pour moi une réelle opportunité, ainsi qu'un fort enrichissement personnel.

1. Plus d'informations ici : http://p4bus.com/
2. Plus d'informations ici : http://llgeometry.com/

3 Découverte

Mes connaissances dans le domaine de la modélisation 3D ainsi que dans l'impression 3D ont nécessité une mise à jour, dans l'objectif d'être capable de cerner au mieux les concepts et enjeux abordés dans le contexte du stage. De nombreuses documentations et recherches ont par ce fait été menées, et ce tout au long du stage. Ces différents travaux sont évoqués dans leur contexte spécifique, permettant ainsi au lecteur néophyte d'aborder sereinement chaque test. Néanmoins, un travail de fond a été effectué lors de la première semaine de stage, sur différents points décrits ci-dessous.

3.1 Géometrie constructive et modélisation solide

Les représentations dites solides [1] font partie d'une branche de la modélisation numérique d'objets, à travers leurs définitions géométriques. Un objet est défini par une combinaison d'opérations sur des objets primitifs tels le cube ou la sphère. Les opérations possibles sont d'une part booléennes : l'union, l'intersection et la différence, et d'autre part des transformations géométriques : translation, rotation, homothétie. On stocke la combinaison d'opérations sous la forme d'un arbre, comme l'illustre la figure suivante. Il est donc possible de définir n'importe quel objet

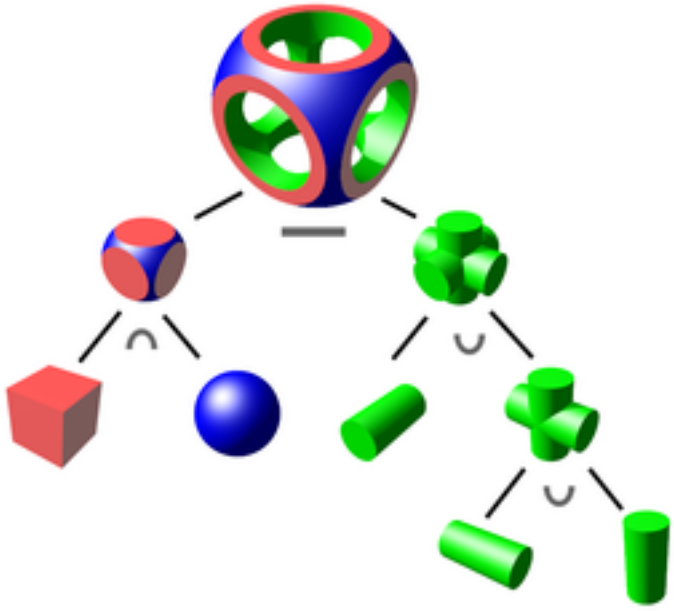

FIGURE 3.1 – Arbre d'opérations

1. Constructive solid geometry (CSG) en anglais

de cette manière, toutefois des objets aux formes complexes peuvent aisément demander de très nombreuses opérations et ainsi alourdir un peu les calculs de rendu. De plus, il n'est pas forcément évident de décomposer l'objet que l'on cherche à représenter, ce qui peut limiter les possibilités de création.

Malgré ces points, cette technique de modélisation propose d'incontournables avantages par rapport à la technique la plus répandue actuellement : la représentation polygonale. Premièrement, les techniques à base de polygones se contentent d'approximer la surface réelle de l'objet, nécessitant de nombreux polygones pour obtenir un résultat visuellement convaincant. La géométrie constructive détient, quant à elle, une définition exacte de l'objet à travers sa formulation mathématique. Autre point important, particulièrement dans le monde de l'impression 3D, l'information de l'intérieur du volume est également stockée. On a ainsi une définition à la fois complète et exacte de l'objet. De plus, les calculs sont largement simplifiés avec cette représentation, qu'il s'agisse de l'intersection avec une droite comme pour le ray tracing ou de projections de formes dans le cas de l'ombrage. Le calcul de collisions devient instantané dans des configurations simples, puisqu'il est possible d'évaluer directement les points que les solides ont en commun. Les gains en volume mémoire sont également impressionnants. En effet, si l'on prend par exemple les deux sphères qui seront utilisées lors de tests ultérieurs, l'espace de stockage nécessaire en polygones est de 1.2Mo, pour un résultat très approximé, dont les facettes sont visibles à l'oeil nu, contre moins d'1Ko en modélisation solide, pour une définition exacte.

3.1.1 Arithmétique des formes

L'arithmétique des formes est un langage universel volumique, permettant la description arithmétique des formes. Elle s'appuie sur les mathématiques pour résoudre des problématiques complexes et ainsi proposer une manière efficace de stocker et transmettre l'information volumique, évitant les excès d'espace mémoire et de temps de calculs réclamés par la représentation polygonale. Le caractère universel de ce langage est basé sur les concepts suivants, énoncés par Jean-François Rotgé dans un acte de conférence :

- Portabilité et indépendance vis à vis du contexte d'utilisation
- Unification qui assure une résolution générale des problématiques géométriques à l'aide d'une algorithmique sans cas particuliers
- Pérénité assurant la validité du langage dans le temps
- Interoptabilité qui assure les futures évolutions contextuelles
- Adaptibilité qui assure les futures évolutions contextuelles
- Compatibilité qui assure l'immigration de l'information déjà disponible

Cette théorie a été inspirée par les travaux de Monge, notamment sa Géométrie Descriptive (1800), ainsi que par l'Algèbre Ornementale de Maurice El-Millick (1936). Sa force est d'intégrer la théorie géométrique fortement abstraite de construction de volumes, le langage topologique. Cela par la transmutation de la problématique topologique et géométrique en problématique logique et arithmétique permettant par la suite de simplifier stockage de l'information et calculs, tout en conservant l'exactitude des résultats.

L'arithmétique des formes englobe la représentation CSG définie auparavant, en corrigeant les problèmes de cette dernière. En effet, les opérations binaires ne sont pas suffisantes pour définir un objet, on est limité à l'état d'appartenance ou non à un objet. Cela a pour conséquence certaines erreurs classiques comme l'effet de peau/pellicule, survenant lorsque l'on fait la différence entre deux objets ayant une surface coïncidente. Ces effets entretiennent une habitude de corruption volontaire des données afin d'obtenir le résultat attendu. Dans cet exemple, il est com-

mun d'étendre l'objet à soustraire pour le faire dépasser de l'objet auquel on le soustrait et ainsi supprimer la zone ambiguë. En arithmétique des formes, les opérateurs N-aires [2] permettent de corriger cela.

L'arithmétique des formes ne sera plus abordée directement lors de ce rapport, mais certains des tests effectués permettront à tous de comprendre la richesse des possibilités offertes par l'arithmétique des formes. Il est difficile d'envisager, du point de vue curieux et intéressé d'un étudiant, les raisons pour lesquelles la représentation polygonale, qui parait préhistorique, tant dans sa conception que dans ses capacités, est toujours prédominante.

3.2 Formats de fichiers

Il est indispensable d'avoir une connaissance au moins globale des formats de fichiers existants pour pouvoir travailler convenablement sur l'un d'entre eux. C'est dans cette optique qu'est proposée cette section.

3.2.1 STL

Le format de fichier STL a été développé par la société 3D Systems. C'est un format de fichiers stockant un objet 3D. Seule la surface de l'objet y est stockée à travers une approximation triangulaire. Son niveau sémantique est très bas, on y définit chaque triangle composant la surface de l'objet représenté, comme l'illustre le listing suivant, qui définit un objet composé d'un seul triangle.

```
solid name
facet normal ni nj nk
   outer loop
     vertex v1x v1y v1z
     vertex v2x v2y v2z
     vertex v3x v3y v3z
   endloop
endfacet
```

Listing 3.1 – Syntaxe STL

Il s'agit bien entendu d'une représentation polygonale, aussi populaire que pauvre en termes de contenu. On devra donc se contenter de l'information de surface, sans couleurs ni définitions de matériaux et même sans unité. Si cela peut sembler léger pour une représentation 3D, ça l'est encore plus quand il s'agit d'imprimer un objet ainsi défini. On peut également noter qu'avec un standard de stockage similaire, la taille d'un fichier dépend intimement de la taille de l'objet à représenter, ainsi que de son niveau de détail.

2. Opérateurs à N dimensions, les opérateurs binaires n'en possédant que deux.

3.2.2 Additive Manufacturing File Format

Après un rapide bilan des possibilités limitées du format STL face aux capacités croissantes des imprimantes 3D, on comprend la nécessité d'établir un nouveau standard de fichiers. L'organisation internationale de normalisation (ISO) ainsi qu'ASTM International ont travaillé de concert sur la conception du successeur de STL, l'Additive Manufacturing File Format (AMF). Les grandes lignes directrices d'AMF sont les suivantes :

1. Indépendant de la technologie
2. Simplicité
3. Evolutivité
4. Performances
   (a) Rapidité de lecture et d'écriture
   (b) Taille de fichier raisonnable
5. Rétrocompatibilité avec STL (Avec perte des données suplémentaires)
6. Extensibilité

AMF est un format basé sur XML, apportant des solutions à certaines lacunes évoquées précédemment. AMF propose ainsi un support natif de la couleur, des matériaux, de la structure interne de l'objet à travers de lattices, ainsi que des constellations permettant de définir plusieurs objets accompagnés de leurs coordonnées dans l'espace au sein d'un même fichier AMF. Les méta-données sont également possibles, mais leur utilisation n'est pas normée. AMF propose aussi la possibilité de définir l'unité de mesure utilisée.

Syntaxe

La syntaxe d'AMF est assez simple, on commence par définir un ou plusieurs objets, contenant une liste de points ainsi qu'un ou plusieurs volumes. Un volume est composé de triangles définis par trois points de la liste précédente. Le listing 3.2 est le code AMF de l'exemple énoncé.

```
<?xml version="1.0" encoding="utf-8"?>
<amf unit="millimeter" version="1.1">
  <metadata type="name">triangle d'example</metadata>
  <object id="3">
    <mesh>
      <vertices>
        <vertex>
          <coordinates>
            <x>1</x>
            <y>1</y>
            <z>0</z>
          </coordinates>
        </vertex>
        <vertex>
          <coordinates>
            <x>1</x>
            <y>0</y>
            <z>0</z>
          </coordinates>
        </vertex>
        <vertex>
          <coordinates>
            <x>0</x>
```

```
            <y>0</y>
            <z>0</z>
          </coordinates>
        </vertex>
      <volume>
        <triangle>
          <v1>0</v1>
          <v2>1</v2>
          <v3>2</v3>
        </triangle>
      </volume>
    </mesh>
  </object>
</amf>
```

Listing 3.2 – Exemple d'AMF

On peut assigner une couleur et un matériau à un volume, définissant respectivement la couleur de la surface et la composition de l'objet comme l'illustre le listing 3.3. Le matériau doit être défini dans le même fichier.

```
<volume materialid="1">
        <color>
          <r>0.8</r>
          <g>0.8</g>
          <b>0.8</b>
        </color>
```

Listing 3.3 – Assignation de couleur et matériau à un volume

La notion de matériau est importante, c'est elle qui permet de définir toute la structure interne de l'objet. Un matériau se définit au même niveau que l'objet. Le listing 3.4 indique la syntaxe de définition d'un matériau.

```
<material id="1">
    <metadata type="name">Materiau A</metadata>
    <color>
      <r>0.8</r>
      <g>0.5</g>
      <b>0.4</b>
    </color>
</material>
```

Listing 3.4 – Définition de matériau

Optimisations

Comme évoqué précédemment, AMF stocke une liste de points, avant de les assigner aux triangles. Cela permet de ne définir qu'une seule fois chaque point, contrairement à STL qui les redéfinit pour chaque triangle. Cela apporte deux avantages, le premier, évident, est une réduction de la taille des données. Le second permet de corriger les erreurs d'ajustements entre triangles de manière native. En STL, cette correction était attribuée aux algorithmes de slicing, chargés de découper l'objet 3D en tranches 2D.

3.2.3 3MF

Le format 3MF est très similaire à AMF, mais il est supporté par un consortium industriel mené par Microsoft (3MF Consortium). Il est nativement supporté par windows 10. 3MF ne fait pas l'objet du stage, mais il en est à l'origine. En effet, c'est l'arrivée de ce concurrent direct à AMF qui a poussé le groupe de travail d'AMF à faire le point sur les capacités actuelles d'AMF. Cela dans le but de pouvoir l'améliorer, et ainsi satisfaire de nouveaux utilisateurs potentiels, tout particulièrement demandeurs sur la représentation solide.

3.2.4 Autres

Il existe de très nombreux autres formats, la liste de ceux étudiés lors de ce stage est disponible dans le tableau 3.2.4.

Format	technologie	commentaires
X3D	polygones	Format prévu pour le web, il contient des informations inutiles pour l'impression comme le son ou l'animation
STEP	modélisation solide	Il est peu utilisé dans la fabrication additive
PLY	polygones	Utilisé pour les données de scanners 3D, support de la couleur
OBJ	polygones	Simple, très utilisé, support de texture mais pas de matériaux
3DS	polygones	Limité à 65636 triangles, support de la texture
ACIS	polygones + modélisation solide	Orienté objet. Très utilisé en conception assistée par ordinateur

4 Audit

Il s'agit ici du coeur du travail réalisé au sein du stage, son objectif étant de cerner les possibilités actuelles du couple AMF/AMFTools. Et ce plus particulièrement concernant la représentation du solide. Pour répondre à ces problématiques, il s'est avéré nécessaire d'adopter plusieurs approches, décrites dans la suite de ce chapitre.

AMFTools est le logiciel open source de référence accompagnant la norme AMF. Il a été développé par Jonathan Hiller. Il comporte déjà de nombreuses fonctionnalités, dont la liste est énoncée sur le wiki d'AMF [2]. Voici les fonctionnalités incluses dans la version 0.9.7 :

- Visualisation et manipulations 3D
- Imports de fichiers STL
- Imports limités de fichiers X3D
- Création et édition des Constellations
- Création, édition et assignement de matériaux à un objet
- Support des équations de répartitions de matériaux à l'intérieur de l'objet
- Vue en coupe de l'objet 3D
- Support des triangles curvilignes
- Interface utilisateur améliorée, avec un accès plus direct aux fonctionnalités.

4.1 Boîte blanche

L'approche boite blanche signifie bien entendu une analyse du code d'AMFTools. Celle-ci s'est avérée nécessaire au début de l'étude globale. En effet, la prise en charge de la modélisation solide par le couple AMF/AMFTools n'étant pas clairement délimitée, la documentation fournie et les exemples mis à disposition des utilisateurs ne permettent pas de créer des fichiers d'exemples. Il est également toujours profitable de connaître le fonctionnement de l'outil que l'on cherche à évaluer.

4.1.1 Recherche d'éléments concernant la gestion du solide

Après un rapide survol du code, il s'est avéré que le support de la modélisation solide n'était pas supporté en tant que tel. Il reste indispensable en AMF d'utiliser un mesh pour décrire la surface du volume. L'objectif de cette sous-section est de montrer les méthodes de recherche choisies ainsi que de présenter les pistes pour les tests ultérieurs.

Libraire MuParser

La présence d'un dossier "MuParser" dans la racine du projet est un début de piste sur le support des lattices et de la représentation fonctionnelle de la structure interne de l'objet. En effet, Muparser est une librairie C++ de parsing d'équations mathématiques, comme l'explique en détail son site [4]. Or la représentation fonctionnelle de l'intérieur de l'objet est une des seules parties d'AMF utilisant potentiellement des équations mathématiques.

Traces d'équations

Par la suite, l'utilisation de l'outil de recherche avancé de SublimeText 3 [1] a permis d'identifier rapidement 7 fichiers dans lesquels il est question d'équations.

Les deux premiers fichiers mentionnés sont Amf.cpp et "Amf_file.cpp", il est question ici à la fois d'équations de mise à l'échelle, qui ne nous concernent pas, et des méthodes permettant de récupérer l'équation d'un composite. Ce qui représente une piste prometteuse.

Le fichier Equation.cpp a tout particulièrement attiré mon attention. Il gère toutes les équations, y compris les équations de mise à l'échelle, mais la partie qui nous intéresse ici se situe dans le parser de l'équation. Plus précisément dans la fonction IniParser(), disponible sur le listing 4.1. On peut constater ici que les équations sont prévues pour contenir des opérations spécifiques à la représentation fonctionnelle : mod, and, or, xor... On remarque également que seules trois variables sont définies, x,y et z. On peut en déduire que la structure interne peut être définie ici en fonction de la position dans l'objet. Nous y sommes, la représentation fonctionnelle est supportée dans la version actuelle d'AMFTools. Reste encore à déterminer à quel niveau elle est implémentée.

```
void CEquation::IniParser()
{
  if (!pP){ //if not already initialized...
    pP = new mu::Parser;

    pP->DefineOprt("%", CEquation::Mod, 6); //deprecated
    pP->DefineFun("mod", &CEquation::Mod, false);
    pP->DefineOprt("&", AND, 1); //DEPRECATED
    pP->DefineOprt("and", AND, 1);
    pP->DefineOprt("|", OR, 1); //DEPRECATED
    pP->DefineOprt("or", OR, 1);
    pP->DefineOprt("xor", XOR, 1);
    pP->DefineInfixOprt("!", NOT);
    pP->DefineFun("floor", &CEquation::Floor, false);
    pP->DefineFun("ceil", &CEquation::Ceil, false);
    pP->DefineFun("abs", &CEquation::Abs, false);
    pP->DefineFun("rand", &CEquation::Rand, false);
    pP->DefineFun("tex", &CEquation::Tex, false);

    pP->DefineVar("x", &XVar);
    pP->DefineVar("y", &YVar);
    pP->DefineVar("z", &ZVar);
  }

}
```

1. SublimeText est un éditeur de texte très complet

Listing 4.1 – parser d'équations

On trouve aussi des traces du mot équation dans la librairie Muparser, ce qui n'est pas une grande surprise, et ne nous concerne pas non plus.

Il reste trois fichiers dans lesquels le terme équation ressort, nComposite, nMaterial et nTexture. Parmi ces trois éléments, nComposite est la clé. En effet, c'est dans le fichier nComposite.cpp que l'on retrouve l'appel du parser d'équations vu précédemment. Un nComposite est un matériau composé de plusieurs matériaux. La documentation d'AMF évoquait la possibilité de créer des matériaux composites avec une proportion d'autres matériaux. On peut faire bien plus, et définir la structure de la matière à travers ce nComposite.

Cette étude du code en "boite blanche" a permis d'identifier comment créer des fichiers de test gérant la répartition de la matière dans l'objet. Il ne reste alors qu'à réaliser ces fichiers de tests, et ainsi démontrer les capacités du couple AMF/AMFTools.

4.1.2 Etude du parser XML

Le parser XML revêt une importance particulière dans AMFTools, puisqu'il s'agit de l'intermédiaire entre le fichier AMF et le reste de l'application. C'est le seul moyen de récupérer les données contenues par le fichier. Ici le parser est couplé au code chargé d'écrire en AMF, dans la classe XMLStream. Les deux outils utilisent la librairie RapidXML [2]. Une gestion des erreurs permet de les remonter dès leur lecture, ce qui est indispensable. Ce parser est appelé à tous les niveaux de l'application, dès qu'une information particulière est requise, comme par exemple la lecture des matériaux. Il convient de noter que le parsing des équations de répartition de la matière n'est pas effectué par ce parser, mais par un parser particulier, présenté dans la sous-section 4.1.1.

4.2 Boîte grise

4.2.1 Étude du Slicer

Définition d'un slicer

Un slicer est un programme permettant de découper un objet 3D en tranches 2D. Dans le domaine de l'impression 3D, il permet, en découpant un objet, d'obtenir des tranches qui seront ensuite converties en instructions d'impression. Une fois une tranche imprimée, on passe à la suivante, et ainsi de suite. Ce qui permet d'imprimer l'objet 3D entièrement à partir d'instructions simples. Dans notre cas, le slicer permet bien évidemment de visualiser ces tranches de la surface, mais il représente surtout le seul moyen de visualiser la structure interne de l'objet, qui est le point central de ce stage. Ce sera donc le seul moyen d'évaluation de nos tests, notre seul retour sur la prise en compte de tel ou tel élément par le couple AMF/AMFTools. Il s'avère donc nécessaire de l'analyser pour en connaître les caractéristiques et spécificités.

2. RapidXML est une librairie très rapide pour le parsing et l'écriture de données XML.

Étude de la classe MeshSlice

Le slicer d'AMFTools est de type voxels. En effet, il parcourt les données du plan xy sous un pas calculé à partir de la précision définie par l'utilisateur, créant ainsi pour chaque tranche une vue à partir d'une matrice de points. C'est un algorithme multi-thread [3]. A l'initialisation, le slicer trie les triangles selon leur plus petite hauteur, en mettant en premier les plus petites hauteurs et en dernier les plus grandes. Pour chaque tranche, il liste les triangles coïncidant avec cette tranche. Par la suite, pour chaque pas sur y, on teste l'intersection entre la droite définie par la valeur de y et les triangles de la liste précédente. A chaque intersection, on remplit les points correspondants avec la couleur du triangle en ce point. Puis, on évalue chaque point de l'intérieur du volume d'après sa définition fonctionnelle. Petite précision, dans le cas d'un point proche de la surface, sa couleur est atténuée, cela aura des conséquences pour nos tests à venir. Puis on répète l'opération jusqu'à avoir parcouru tout l'objet.

Sans être expert du domaine, il paraît évident ici que le traitement de la dualité surface/structure interne induit de nombreux calculs. En effet, on doit déjà trier les éléments de la surface, tester lesquels sont dans le plan sur lequel on travaille, puis dans une autre opération, calculer l'intersection de tous les points de chaque élément de la surface avec le plan. Ensuite, pour chaque point interne, il faut déjà s'assurer qu'il est bien à l'intérieur de la surface (ce qui représente une opération assez coûteuse). Enfin, on calcule la fonction de répartition pour chaque point interne.

Comme nous le remarquerons à travers un test dans la sous-section 4.3.2, il est tout à fait possible de s'affranchir de la surface, sans perdre de données (au contraire). Ce qui ramènerait cette montagne d'opérations à une seule intersection entre le plan et l'objet. Il existe des slicers capables de gérer aussi bien le solid modeling seul que combiné avec des polygones, comme le très complet IceSl développé par Sylvain Lefèbvre [3].

4.2.2 Filières de conversion d'images

AMF prévoit le support des textures à la fois sur la surface externe, ainsi que dans le cadre de la définition de la structure interne de la matière. Dans cette section, nous n'aborderons que l'ajout d'une image dans le format adapté à AMF, son utilisation étant détaillée plus tard, dans la sous-section 4.3.5.

Syntaxe

La documentation de la norme AMF[8] précise que l'image à utiliser pour texturer doit être un tableau d'octets, représentant chacun une valeur de 256 niveaux de gris correspondant à un pixel. On retrouve dans le listing 4.2 la syntaxe AMF. Prenons une image composée uniquement de niveaux de gris. Il paraît assez évident de stocker une texture de cette manière : on lui attribue un identifiant, on définit sa hauteur, sa largeur et éventuellement sa profondeur. On remarque aussi qu'il y a un type à définir, celui-ci sera toujours "grayscale" pour correspondre aux niveaux de gris définis par la norme.

```
1 <texture id="1" width="2" height="2" depth="1" tiled="0" type="grayscale">
    hQAAhQ==</texture>
```

Listing 4.2 – syntaxe balise texture

3. un Thread est un fil d'exécution, utilisé pour accélérer le traitement de manière pseudo-parallèle

Préparation des données

Enfin on peut voir à l'intérieur de la balise de texture, notre texture. Cette partie est moins évidente. Bien que la norme précise que l'encodage utilisé est Base64 [4] et qu'il faut encoder un tableau de pixels sans méta-données, il n'est pas évident pour un néophyte de réaliser cette conversion. La première étape de cette conversion consiste à débarrasser notre image, et ce quel que soit son format d'origine, des méta-données quelle comporte. Pour cela, le plus simple est d'ouvrir notre image dans un logiciel de traitement d'image. Dans le cadre du stage, c'est FiJi [5] qui sera utilisé, mais tout logiciel permettant un export dans le format .raw, exempt de toute méta-donnée, fonctionne également.

Encodage en Base 64

Le format raw contient donc notre tableau de pixels, il nous faut maintenant l'encoder en Base64. Une librairie c/c++, libb64 [6], permet l'encodage et le décodage de données en Base64, via une simple commande console sous Windows ou Linux. Une fois les données encodées en base 64, il suffit de les insérer dans la balise texture comme précédemment.

Cas d'une image couleur

Nous avons vu comment importer une image en niveaux de gris. Pour une image couleur, il faut auparavant séparer les différents canaux de couleurs et éventuellement celui de transparence avant d'effectuer la manipulation précédente pour chacun d'entre eux.

Pistes d'automatisation

La manipulation globale nécessaire pour ajouter une texture dans un fichier AMF est donc loin d'être évidente, c'est pourquoi il serait bénéfique d'ajouter une filière d'import de textures à la librairie d'AMFTools. L'import de tous types de fichiers est compliqué à mettre en oeuvre, et finalement n'est pas du ressort d'AMFTools. Il est possible en revanche d'automatiser le processus d'intégration d'un format d'image en particulier. Des recherches ont été réalisées dans ce sens, et mènent vers plusieurs pistes. Le format PNG [7] correspond à nos besoins, car au-delà de sa très forte utilisation, il propose une compression sans perte, un stockage matriciel des données ainsi que la gestion de la transparence. De plus, après une courte étude, la librairie c/c++ libpng dispose de tous les outils pour séparer les canaux de couleurs, tout en permettant de ne garder que le tableau de pixels dont nous avons besoin. En couplant cette sortie à libb64 présentée précédemment, nous obtiendrions une conversion automatisée. Il ne resterait alors plus qu'à écrire la texture dans le fichier AMF via la Classe XMLstream [8].

4. Base64 est un codage de l'information utilisant 64 caractères, principalement utilisé pour la transmission sur internet.
5. Fiji is Just imaji est un logiciel de traitement d'image libre très complet.
6. Libb64 est disponible à cette adresse : http ://sourceforge.net/projects/libb64/
7. PNG signifie Portable Network Graphics
8. Élément actuel de la suite AMFTools permettant les opérations d'écriture dans un fichier AMF

4.3 Boîte Noire

4.3.1 Utilisation des matériaux pour structurer l'intérieur de l'objet

Comme vu précédemment dans la sous-section 4.1.1 ,c'est dans la définition de matériaux, plus précisément de composites, que se trouve la possibilité de définir la structure interne de l'objet. Pour comprendre la syntaxe à adopter, il a fallu compiler des éléments de différentes sources. Il s'agissait tout d'abord de déterminer comment stocker une équation selon la norme XML, afin que celle-ci ne soit pas parsée par un parser XML. Il suffit pour cela d'utiliser la balise CDATA, dont l'usage est illustré par le listing 4.3

```
<![CDATA[ notre équation ]]>
```

Listing 4.3 – balise CDATA

La syntaxe de l'équation en elle-même se déduit de deux éléments. Elle est présentée dans une lettre publiée par J. Hiller et H. Lipson [7] qui a servi de base à la construction d'AMF. Elle a également été identifiée dans l'étude du code, plus précisément dans le listing 4.1.

Enfin, la manière d'inclure ce composite dans un matériau a été déterminée en utilisant AMF-Tools, dans la partie dédiée à l'édition des matériaux. La syntaxe globale est décrite par le listing 4.4. Remarque : nous parlerons ici de proportions de matériaux, telles que présentées dans la norme, mais il s'agit d'un abus de langage car chaque matériau se voit assigner une valeur entre 0 et 1, indépendamment des autres matériaux composant le matériau principal.

```
<material id="1">
    <metadata type="name">matROUGE</metadata>
    <color>
      <r>0.666667</r>
      <g>0</g>
      <b>0</b>
    </color>
  </material>
  <material id="2">
    <metadata type="name">Composite</metadata>
    <color>
      <r>0</r>
      <g>0</g>
      <b>0</b>
    </color>
    <composite materialid="1"><![CDATAé[\quation de ré\partition]]></composite
    >
</material>
```

Listing 4.4 – syntaxe matériau composite

Toujours dans l'exemple du listing 4.4, il suffit de remplacer par l'équation choisie, celle-ci définissant la proportion de la matière correspondant au materialid défini. Il est possible de cumuler ses définitions de composites, afin de stocker de nombreuses informations dans un même matériau.

4.3.2 CSG Frep

Comme décrit précédemment, AMF ne permet pas de définir des objets en CSG. Néanmoins il est possible de définir l'intérieur des objets de manière fonctionnelle. C'est un outil très puissant, qui permet de stocker à moindre coût mémoire de nombreuses informations. On peut ainsi conserver les intentions de design, en gardant une trace de la construction de l'objet, et ce de manière très simple. Il suffit en effet d'attribuer un matériau pour chaque objet composant l'ensemble. Il suffit pour cela de définir ces objets par leurs équations mathématiques et d'attribuer un matériau spécifique à chaque équation.

Environnement de test

Afin de mettre en évidence la capacité d'AMF à conserver les intentions de design, un test simple à base de deux sphères A et B est imaginé. Ces deux sphères se coupent, elles ont donc une partie commune. L'idée est d'arriver à définir dans l'intérieur du matériau trois zones : $A - A \cap B$ qui sera représentée en rouge, $A \cap B$ en bleu et enfin $B - A \cap B$ en vert.

Deux étapes sont prévues : la première consiste à créer les équations de ces deux sphères sans prendre en compte les éventuelles perturbations engendrées par la surface. Or il n'est pas possible de créer un fichier AMF sans surface, nous allons donc nous créer un "bac à sable" dans un cube plus grand que les deux sphères. Dans le cas où ce test s'avère concluant, on pourra tester les mêmes formules internes, mais cette fois-ci dans un mesh contenant la surface extérieure de $A \cup B$. La figure 4.1 permet de visualiser en 3D les environnements de test.

Remarque : Les objets ont été générés de manière à avoir toute coordonnée positive.

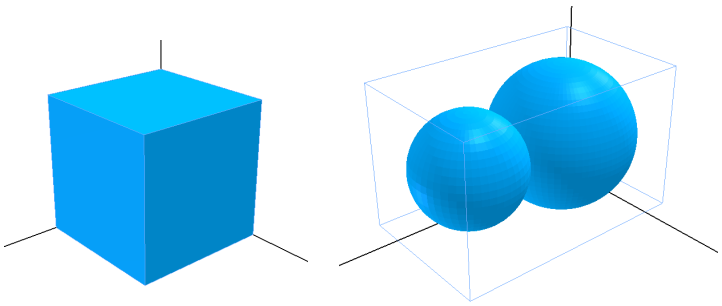

FIGURE 4.1 – Surfaces dans lesquelles seront effectués les tests

Implémentation

L'implémentation est définie dans le listing 4.4. Il suffit d'insérer l'équation correspondant à nos objets géométriques, ici nos deux sphères. L'équation d'une sphère de centre $M(a, b, c)$ et de rayon R s'écrit $(x-a)^2 + (y-b)^2 + (z-c)^2 = R^2$. Nommons A l'équation de notre sphère A et B l'équation de notre sphère B, les coordonnées réelles ne nous intéressent que peu ici, nous nous en abstrairons donc. L'opération de différence n'étant pas disponible, il faut la réaliser avec une combinaison d'opérations disponibles. Donc pour réaliser $A - A \cap B$ par exemple, il faudra utiliser l'équation $(A\ XOR\ B)\ AND\ A$.

Résultats

Le premier test, privé des éventuels conflits avec les surfaces, s'est avéré concluant. En regardant les différentes tranches 2D obtenues à l'aide du slicer de la figure 4.2, on distingue bien la répartition de la matière dans les deux sphères et l'intersection de ces deux dernières. Le deuxième test

FIGURE 4.2 – Tranches 2D de bas en haut du cube dans lequel on a défini nos deux sphères

disponible en figure 4.3 corrobore la possibilité de sauvegarder les intentions de design et plus largement, la structure interne de l'objet via une représentation fonctionnelle. On peut voir ici, contrairement au premier exemple, un fin liseré noir qui représente la surface de l'objet.

FIGURE 4.3 – Tranches 2D de bas en haut des deux sphères

Un oeil observateur remarque la redondance des données ici. En effet, si l'on compare les deux tests, il est aisé de se rendre compte que l'on retrouve les même informations. Or, souvenez-vous, lors du premier test, nous avions totalement occulté l'information de surface. Il devient alors évident que toute l'information peut être portée par la définition arithmétique de l'objet, et que l'on peut aisément se passer de la surface polygonale pour définir l'objet. Cela permettrait d'économiser 1.2Mo d'espace (pour un fichier d'1.2Mo, qui ferait alors quelques centaines d'octets), ainsi que de nombreux calculs effectués au niveau du slicing, souvenez-vous de la sous-section 4.2.1

4.3.3 Lattices

Un peu de théorie...

Une lattice est un ensemble partiellement ordonné d'éléments. En d'autres mots, une lattice définit la répartition des matériaux dans un objet. Étant néophyte dans ce domaine, des recherches supplémentaires ont été nécessaires, afin d'assurer une meilleure compréhension des intérêts et des propriétés de ces lattices. La consultation d'un livre [5] sur la théorie des lattices a été très formatrice. Il existe des exemples connus de lattices, notamment dans les théories de pavage [9] de l'espace que l'on peut visualiser sur la figure 4.4. Chaque lattice a des caractéristiques propres, la

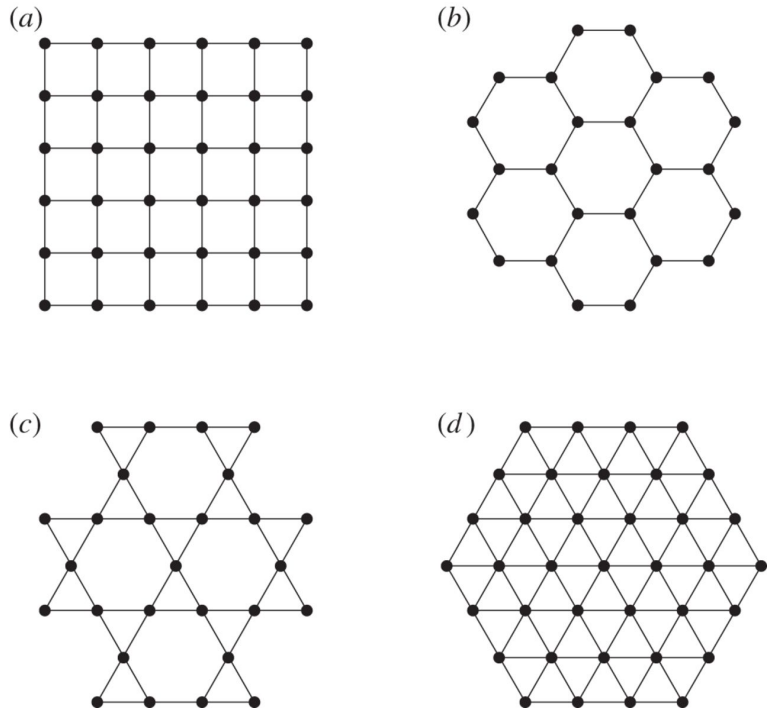

FIGURE 4.4 – Exemples de lattices 2D A : carrée, B : nids d'abeille, C : kagomé, B : triangulaire

rendant plus adaptée à une application particulière. Prenons la lattice en nids d'abeille en exemple, qui définit le pavage 2D le plus efficace, ce fait ayant été démontré par T. C. Hales dans sa publication nommée "HoneyComb Conjecture" [6].

Nous pouvons étendre ces considérations d'optimisation en 3D. Le pavage parfait à l'image du nid d'abeille en 2D n'est pas encore connu. Seulement quelques théoriciens se rapprochent de

9. Remplissage de l'espace selon un schéma défini

cet idéal à travers leurs propositions, tel Kelvin et son octoèdre tronqué. D'autres chercheurs ont trouvé des pavages semi-réguliers plus efficaces, mais de peu.

Néanmoins malgré ces considérations, Jérémie Farret indique que selon les technologies d'impression utilisées, de telles théories peuvent devenir non optimales voire problématiques, selon l'orientation des strates de dépôt de matériau, par exemple, ou encore les effets d'aliasing (Crénelage) sur les données d'entrée. Ces perturbations peuvent entraîner une forte variation des caractéristiques essentielles des éléments imprimés, comme leur rigidité . La lattice cubique étant largement utilisée et plus simple à implémenter, elle fera l'objet de nos tests.

Après cette brève description des lattices, on comprend vite l'intérêt de pouvoir stocker ces lattices dans le même fichier que la surface, les deux formant un tout. Cela permet de conserver les propriétés initiales de l'objet, et ainsi le stocker comme il a été conçu. C'est pour cette raison qu'il était important de déterminer dans quelle mesure les lattices étaient supportées, et comment les utiliser.

Environnement de test

Il était important de tester l'implémentation des lattices dans de nombreuses configurations d'objets, dans le but d'identifier des erreurs éventuelles. Ces erreurs peuvent être dues à un objet creux ou concave par exemple. Il était également important de représenter les origines possibles [10] des objets. Différents objets feront donc partie de notre batterie de test.

— Un cube défini directement en AMF visible sur la figure 4.5

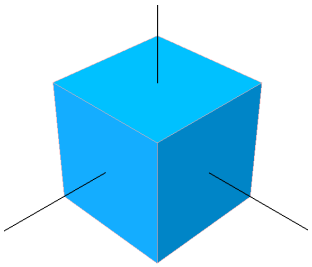

FIGURE 4.5 – Vue 3D du cube de test

— Une colonnade réalisée par mes soins sous OpenSCAD [11] puis exportée vers AMF depuis OpenSCAD. (figure 4.6).

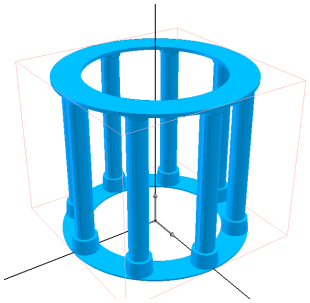

FIGURE 4.6 – Vue 3D de la colonnade de test

— Une réalisation nommée "CircleThing" postée sur ThingInverse [12], qui est suffisamment complexe pour effectuer de nombreux tests en un. C'est à l'origine une conception sous OpenSCAD, exportée en fichier STL puis importée en AMF sous AMFTools, elle est illustrée par la figure 4.7.

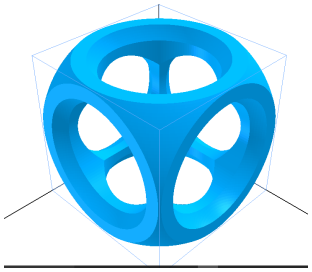

FIGURE 4.7 – Vue 3D de l'objet "circlething" de test

10. OpenJscad, site web libre de modélisation 3D, mis en avant dans les consignes du stage, propose un export vers AMF particulier, avec un volume par triangle, ne permettant pas de définir la structure interne de l'objet.
11. OpenSCAD, logiciel libre de modélisation 3D, est basé sur la modélisation constructive définie section 3.1
12. ThingInverse est un site de partage de fichiers STL lancé par MakerBot

Implémentation

L'implémentation suit celle définie dans le listing 4.4. Il suffit d'insérer l'équation correspondant à notre lattice. Il s'avère aisé de définir une lattice cubique grâce à l'opération modulo. Cela donne l'équation suivante :
$(mod(abs(x), 10) < 2) or (mod(abs(y), 10) < 2) or (mod(abs(z), 10) < 2)$
Intéressons-nous à la première des trois composantes séparées par un "OU" logique :
$mod(abs(x), 10) < 2)$.
Simplement, lorsque la valeur absolue de x est proche d'un multiple de 10, cette partie renvoie 1 et partout ailleurs 0. On assignera donc 1 à la "proportion" de matière rouge à cet endroit. Ce qui donne une suite de lignes parallèles, toutes perpendiculaires à l'axe des abscisses. En ajoutant avec un OU logique la même équation, mais cette fois en y, on se retrouve avec un quadrillage donc une lattice carrée. Enfin, en ajoutant z de la même manière, on obtient une lattice cubique.

Résultats des tests

Procédons par étapes, regardons dans un premier temps ce que le test sur le cube a donné. Afin de visualiser le résultat, nous n'avons d'autre choix que de montrer les tranches [13] visualisées à partir du slicer d'AMFTools. Dans le cas du cube illustré en figure 4.8, on retrouve bien notre lattice cubique, avec sur l'image de gauche la tranche lorsque z n'est pas proche d'un multiple de 10. Et à droite la tranche quand z est proche d'un multiple de 10, formant ainsi une couche de matière uniforme. Le test est donc concluant.

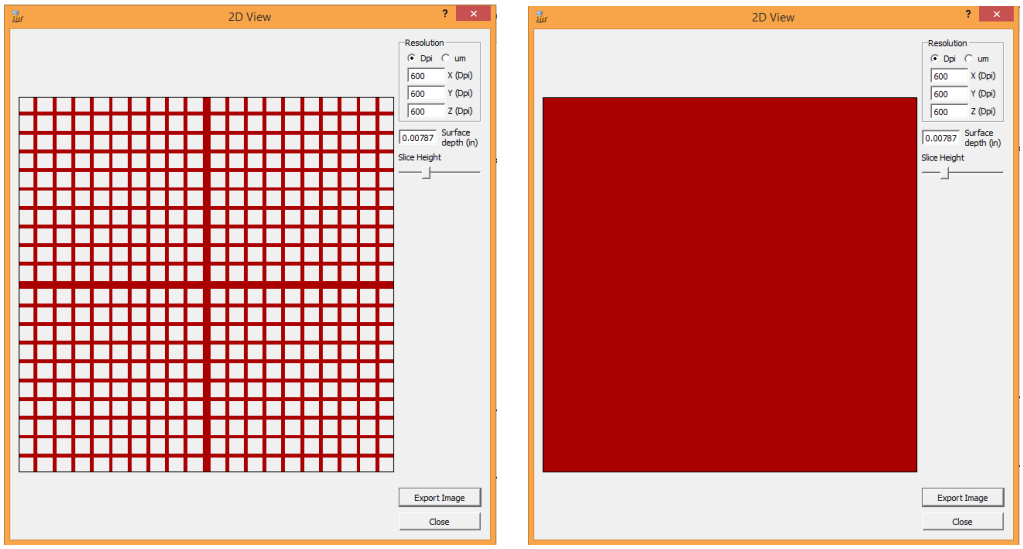

FIGURE 4.8 – Tranches 2D du cube rempli avec une lattice cubique

Intéressons-nous maintenant au remplissage de notre colonnade tel qu'illustré par la figure 4.9. La tranche de gauche étant réalisée au niveau des colonnes, tandis que la droite se situe au niveau de l'anneau au sommet des colonnes. une fois de plus, on retrouve notre lattice, correctement définie dans l'objet, le test est également concluant.

13. Tranche : coupe en 2D pour une hauteur fixée de l'objet

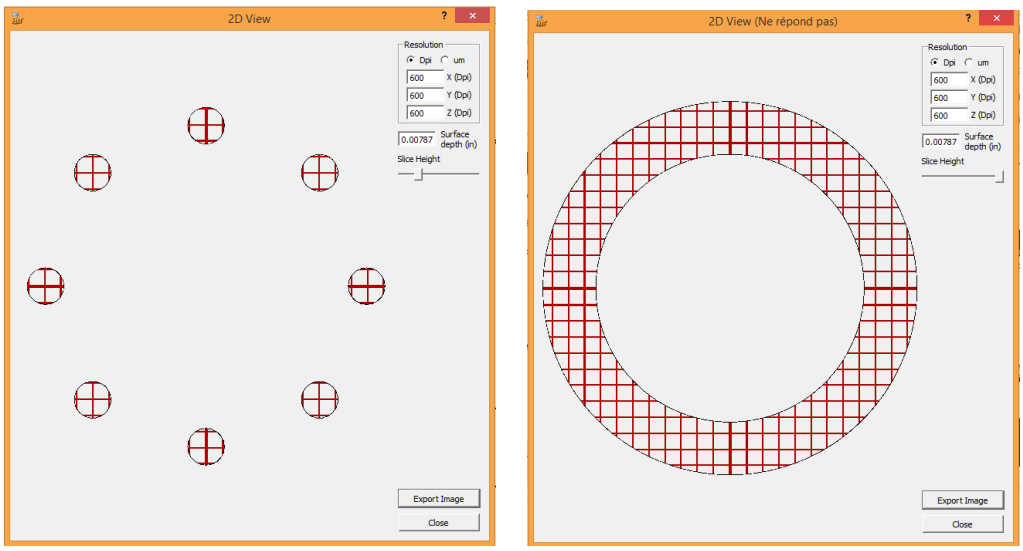

FIGURE 4.9 – Tranches 2D de la colonnade remplie avec une lattice cubique

Dernier élément, certainement l'exemple le plus significatif, notre "circlething". On peut parfois voir des bavures, engendrées par le slicer, mais elles paraissent aléatoires. Malgré ce défaut du slicer, les résultats obtenus, illustrés par la figure 4.10, confirment les résultats précédents et permettent d'affirmer que les lattices sont bel et bien utilisables dans la version actuelle du couple AMF/AMFTools.

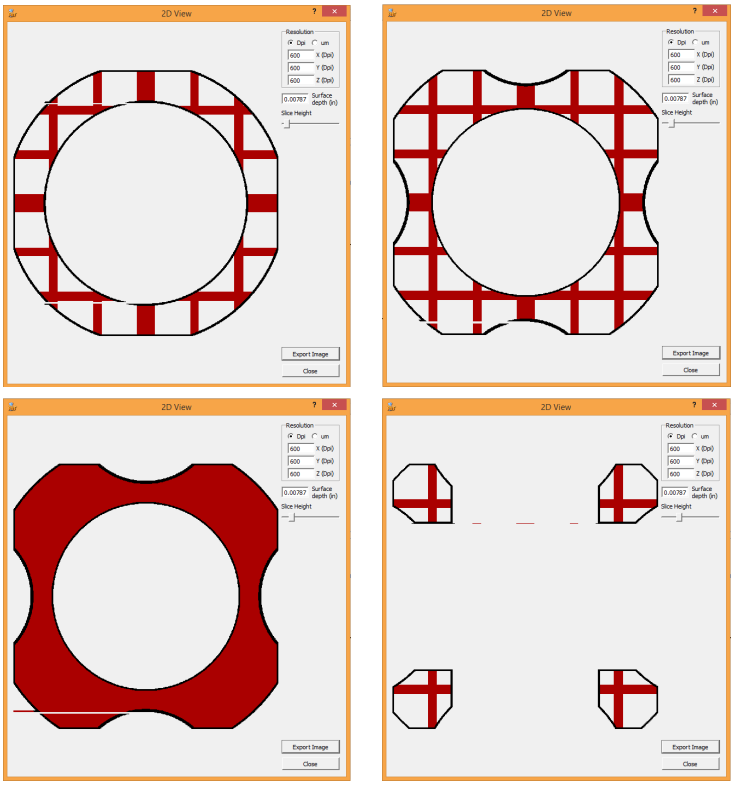

FIGURE 4.10 – Tranches 2D de "circlething" remplie avec une lattice cubique

4.3.4 Représentation de données voxel en AMF

L'utilisation de voxels [14] explosant ces dernières années, il est intéressant pour une norme comme AMF d'être capable de l'intégrer. Certains acteurs du workgroup AMF utilisent des données voxels dans leurs domaines respectifs, plus particulièrement dans le domaine de la santé, pour de nombreux type d'imagerie médicale, tel la tomographie [15]. Deux approches ont été dégagées pour stocker des données voxels en AMF. La première, émise par le Workgroup AMF, consiste à créer un volume pour chaque voxel. La seconde, imaginée à partir des travaux déjà réalisés au sein du stage, consiste en l'export de la surface de l'objet en voxels et le stockage des voxels internes via la représentation fonctionnelle.

Approche Multi-volumes

Il s'agit ici d'exploiter l'outil constellation proposé par AMF. Le concept est simple, il suffit de définir nos différents objets dans un premier temps. Dans notre cas, nous n'aurons que des cubes 1x1x1, mais il faut en définir un pour chaque matière différente que l'on souhaite représenter. Par la suite on définit une constellation [16] comme l'illustre le listing 4.5, en créant des instances de chaque objet, copie conforme de l'objet défini, mais en précisant sa position ainsi que sa rotation dans l'espace.

```
<constellation id="3">
  <metadata type="name">Constellation principale</metadata>
  <instance objectid="1">
    <deltax>0</deltax>
    <deltay>0</deltay>
    <deltaz>0</deltaz>
    <rx>0</rx>
    <ry>0</ry>
    <rz>0</rz>
  </instance>
  <instance objectid="2">
    <deltax>2</deltax>
    <deltay>0</deltay>
    <deltaz>0</deltaz>
    <rx>0</rx>
    <ry>0</ry>
    <rz>0</rz>
  </instance>
</constellation>
```

Listing 4.5 – syntaxe d'utilisation d'une constellation

Il est également possible d'affecter une texture à la surface de notre cube, ce qui a été réalisé dans le cadre de l'exemple, mais cette partie est suffisamment documentée dans la norme[8] et ne sera pas traitée dans ce rapport. Le logo d'AMF étant une tour, c'est cette forme qui a été choisie dans le cadre de notre test dont le résultat est visible sur la figure 4.11

14. Un voxel est un pixel en 3D, on lui attribue une position dans l'espace, ainsi qu'un nombre non borné de propriétés

15. technique de numérisation de données non intrusives, consistant en la mesure tranche par tranche, avant de reconstruire l'objet analysé en empilant ces tranches

16. Les identifiants sont partagés avec les éléments d'autres types, il convient donc d'assigner un id unique à cette dernière.

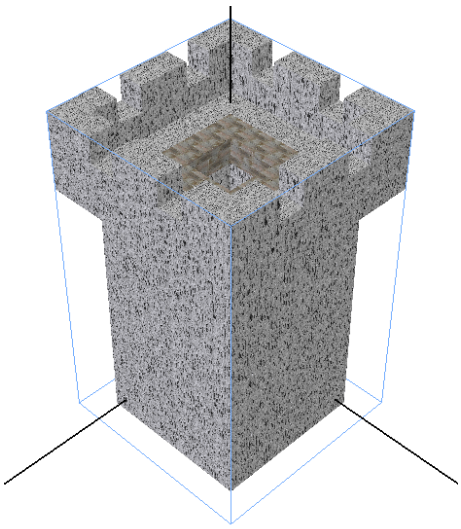

FIGURE 4.11 – Vue 3D de la tour voxel

D'après la vue 3D, il pourrait tout à fait s'agir d'un seul objet. La vue en tranches de la figure 4.12 permet néanmoins de visualiser sa structure.

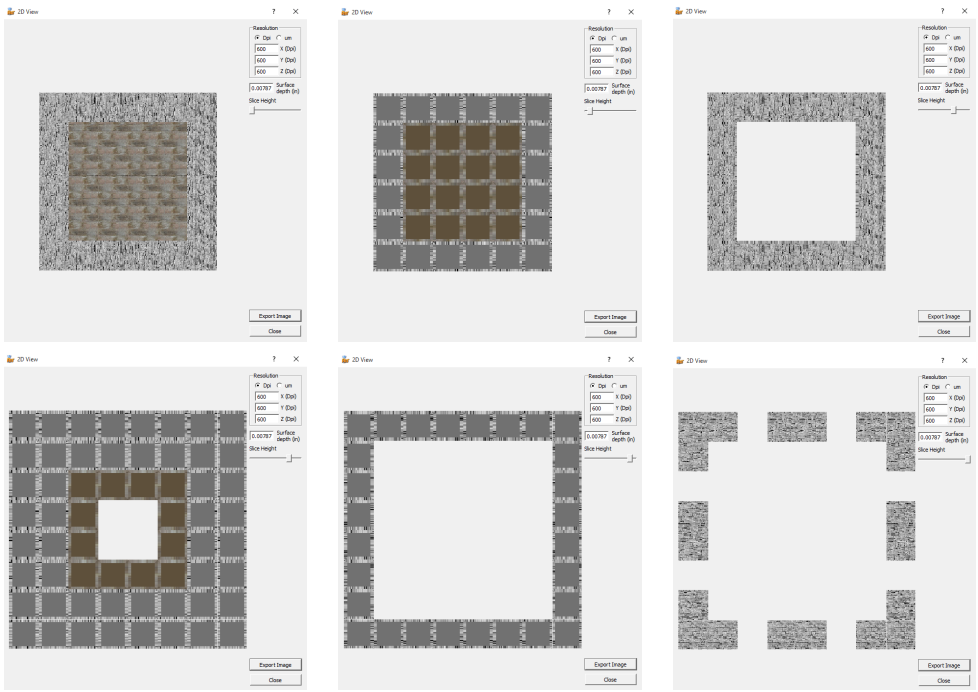

FIGURE 4.12 – Tranches de bas en haut de la tour voxel

Ces tests prouvent donc qu'il est possible de stocker des données voxels dans le format AMF, sans perte de données. Cela s'accompagne des caractéristiques des données voxels, particulièrement leur volume. Plus l'objet est grand et plus la précision est grande - plus les voxels sont petits - plus la taille mémoire de l'objet va augmenter. On peut facilement se retrouver avec des fichiers de tailles vraiment conséquentes. Il existe une optimisation classique, consistant à ranger nos pixels sous formes d'octree [17]. On construit un arbre qui subdivise chaque cube en 8 s'il contient des variations. En commençant par un cube contenant tous les voxels, en subdivisant afin de ne

17. optimisation de l'espace permettant parfois de réduire drastiquement le nombre de voxels nécessaires à la représentation de l'objet.

perdre aucun détail. En d'autres termes, si huit cubes identiques forment un cube, on stockera uniquement ce cube. Il est possible de représenter des données voxels optimisées par Octree, mais paradoxalement, même si cela réduit le nombre de voxels enregistrés, cela nécessite la définition d'autant de meshs que de niveaux de subdivision de l'octree. Ce qui peut représenter de nombreux octets.

Approche fonctionnelle

L'idée est d'exploiter la représentation fonctionnelle précédemment démontrée afin de modéliser les voxels à l'intérieur de la surface polygonisée de l'objet voxel. Afin de faciliter la génération d'un exemple convaincant, l'outil en ligne proposé par 3DSlash[1] a été utilisé. Cet outil permet de réaliser facilement un objet en ajoutant/enlevant des voxels à une cuve de voxels. La figure 4.13 permet de visualiser l'objet de test. Il est possible également d'appliquer une texture à cette surface pour correspondre au rendu des voxels internes, mais cela sort du cadre de notre test.

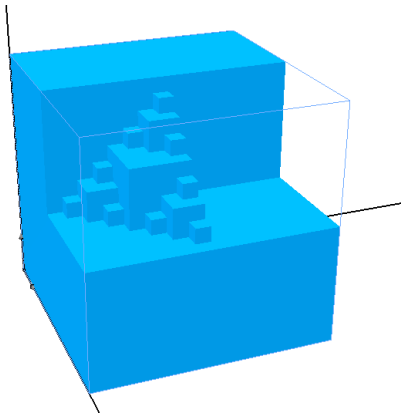

FIGURE 4.13 – Vue 3D de la surface de test

La définition fonctionnelle de chaque voxel est simple, il est délimité par 6 plans, coïncidents avec ses faces. Ce qui donne concrètement comme équation :

$$x > Xmin \quad AND\ x < Xmax\ AND\ y > Ymin\ AND\ y < Ymax\ AND\ z > Zmin\ AND\ z < Zmax$$

Pour l'implémenter, il suffit de placer cette équation dans l'emplacement prévu à cet effet dans le listing 4.4 et d'assigner la matière désirée en choisissant le materialid correspondant. C'est simple et cela permet de représenter des données voxels optimisées par octree sans coût de volume de fichier supplémentaire.

Le résultat du test sur la figure 4.14 met bien en avant la possibilité d'optimisation par octree via les voxels de différentes tailles. Cette approche comporte une certaine limitation, se situant dans le temps de calcul nécessaire au slicer. En effet, comme vu précédemment, le slicer étant de type voxel, la complexité du calcul final peut poser problème lors du traitement de grandes instances.

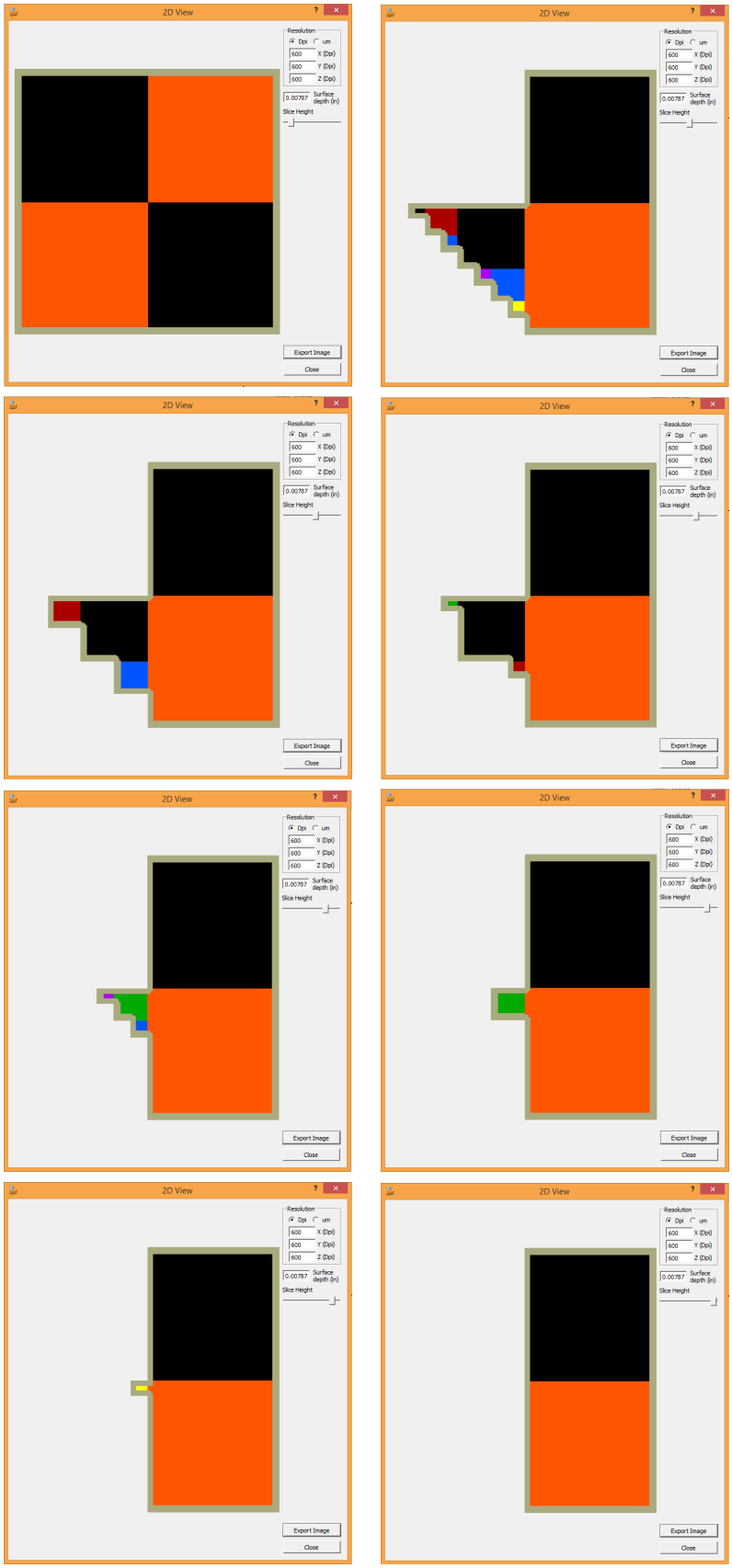

FIGURE 4.14 – Tranches de bas en haut de notre objet

4.3.5 Texturing 3D

L'application d'une texture sur la surface externe est connue et documentée, elle ne fait donc pas l'objet de ce stage. En revanche l'utilisation de texture 2D et 3D pour définir la structure interne de l'objet nous intéresse tout particulièrement. L'intégration d'une image dans un fichier AMF a été traité dans la sous-section 4.2.2, nous n'y reviendrons donc pas. L'objectif est ici de stocker l'information de répartition de la matière dans une texture et d'arriver à utiliser cette information.

Environnement de test

Il est possible d'effectuer ce test dans n'importe quelle forme de surface, mais dans un premier temps, dans le but de simplifier l'expérience, nous nous limiterons à des cubes de différentes tailles. Coté textures, notre ensemble de tests visible en figure 4.15 a été converti en textures AMF selon la méthode décrite dans la sous-section 4.2.2. L'idée de ce panel est de tester toutes les textures sur des cubes de différentes tailles, afin de voir de quelle manière AMFTools réagirait. Les cubes sont de tailles 1*1*1,2*2*2,10*10*10,32*32*32,100*100*100 et ont été générés à partir du cube 1*1*1 déjà utilisé dans d'autres tests dont le visuel est disponible sur la figure 4.5. Ici l'unité définie est le mm. Après validation des tests avec des textures 2D, des tests utilisant des textures 3D seront mis en place.

FIGURE 4.15 – Textures de test

Implémentation

On utilise toujours ici la représentation fonctionnelle pour définir la structure interne de notre objet, définie dans la section 4.3.1. On s'intéressera ici plus particulièrement à la fonction tex(texid,u,v,w) disponible dans AMF. Cette fonction, d'après sa description dans la lettre d'ouverture d'AMF[7], renvoie une valeur entre 0 et 1 correspondant au pixel de coordonnées u, v et w. Après quelques tests préliminaires, il a été déterminé que l'indexation des textures allait de 0 à $NombrePixels - 1$. Les coordonnées u, v, et w que l'on doit fournir sont des entiers. Comme le montre la section 4.2.1, le slicer ne prenant pas en compte les éléments internes proches de la surface, il n'est pas nécessaire de se prémunir contre un dépassement en enlevant la valeur maximale. Dans ce cadre, pour obtenir u en fonction de x, tout en prenant en compte la taille de la texture ainsi que celle de l'objet à texturer, on obtient la formule suivante : $u = floor(x * \frac{Nombre\_de\_pixels}{Taille\_objet})$
Ce qui donne pour l'équation de répartition de la matière :
$tex(1, floor(x * \frac{Nombre\_de\_pixels}{Taille\_objet}), floor(y * \frac{Nombre\_de\_pixels}{Taille\_objet}))$
Remarquez ici que l'id de la texture est 1, ce qui correspond dans notre exemple à la texture en niveaux de gris représentant le rouge. Il convient donc d'assigner un matériau rouge à cette équation.

Si l'on prend par exemple une texture 10*10 et un cube 2*2*2 cela donne le code AMF du listing 4.6, les textures 1, 2 et 3 correspondant respectivement aux niveaux de rouge, vert et bleu.

```
<material id="1">
  <metadata type="name">R</metadata>
  <color>
    <r>1</r>
    <g>0</g>
    <b>0</b>
  </color>
</material>
<material id="2">
  <metadata type="name">G</metadata>
  <color>
    <r>0</r>
    <g>1</g>
    <b>0</b>
  </color>
</material>
<material id="3">
  <metadata type="name">B</metadata>
  <color>
    <r>0</r>
    <g>0</g>
    <b>1</b>
  </color>
</material>
<material id="4">
  <metadata type="name">Texture</metadata>
  <color>
    <r>0</r>
    <g>0</g>
    <b>0</b>
  </color>
  <composite materialid="1"><![CDATA[tex(1,floor(x*10/2),floor(y*10/2))]]></composite>
  <composite materialid="2"><![CDATA[tex(2,floor(x*10/2),floor(y*10/2))]]></composite>
  <composite materialid="3"><![CDATA[tex(3,floor(x*10/2),floor(y*10/2))]]></composite>
</material>
```

Listing 4.6 – Implémentation de la texture dans le solide

Résultats des tests

L'application de chaque texture à chaque cube a été effectuée dans le cadre de ces tests. Nous ne présenterons en revanche ici qu'un condensé des visuels obtenus, en éliminant les informations redondantes, ne conservant ainsi que les éléments significatifs. La première série de tests s'est effectuée sur la texture 1*1, que l'on nommera ici texture 1. Les cubes seront également référencés par leur côté, par exemple cube 10 pour un cube de 10*10*10. La figure 4.16 montre les tranches obtenues lors de ce permier test. On peut voir que dans le cas des cubes 1 et 2, auxquels on applique la texture 1, cela fonctionne, à un écart de couleur près, explicité lors de l'étude du slicer dans la sous-section 4.2.1. Mais si l'on se penche sur les deux résultats de droite, on pourrait croire à un nouvel écart de couleur, mais si c'était le cas la couleur serait plus fade en raison des modifications volontaires appliquées par le slicer. Il s'agit donc ici d'une erreur du slicer.

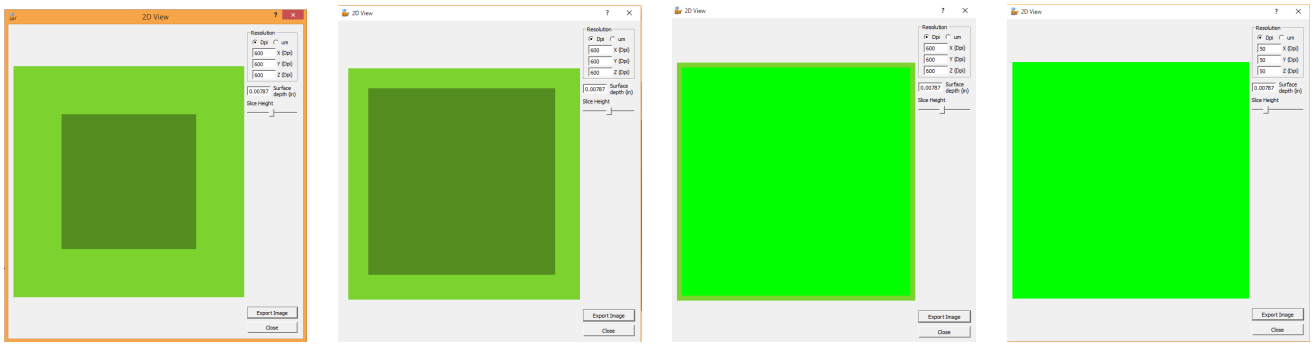

FIGURE 4.16 – Application de la texture 1 aux cubes 1,2,10,256

Prenons maintenant le cube 1 et appliquons-lui différentes textures. Le résultat obtenu, disponible en figure 4.17, montre un résultat positif dans le cas de la texture 2, mais des résultats totalement incohérents pour les textures 10 et 32. On semble ici avoir affaire à des erreurs d'accès mémoire. La fonction tex() n'est donc pas fonctionnelle en l'état.

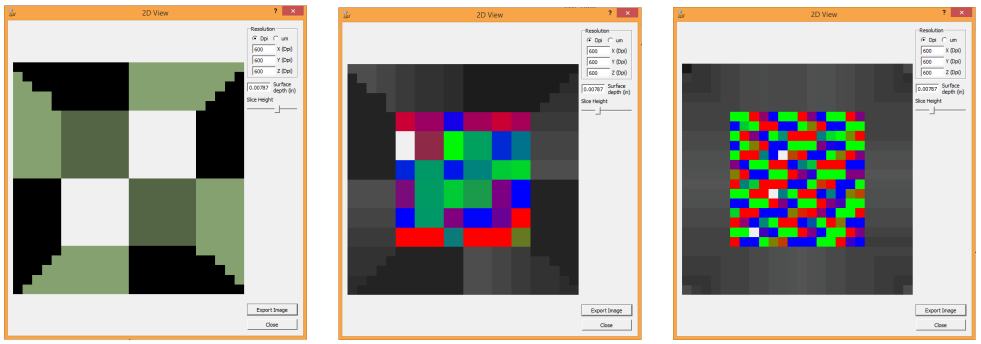

FIGURE 4.17 – Application des textures 2, 10 et 32 au cube 1

Suite aux résultats de ces tests, et la correction de bugs ne faisant pas partie des objectifs du stage, il a été décidé d'envoyer un rapport d'erreur au groupe de travail AMF. Ce rapport d'erreur a été réalisé par Jérémie Farret, sur la base des travaux de ce stage.

4.3.6 Support polygonal de sources solides

L'objectif de cette section est d'identifier la capacité d'AMF à contenir des conceptions solides initialement réalisées en CSG ou en arithmétique des formes. Cela concerne plus précisément des fichiers complexes, composés de plusieurs volumes ayant des propriétés propres. Il est donc question d'analyser le comportement d'AMF lorsque des objets sont ajustés avec précision, avec par exemple des surfaces en contact.

Environnement de test

L'objet servant de base à nos test est la vasque Frank Lloyd Wright [18], modélisée en arithmétique des formes. On peut la voir ainsi que ses éléments constitutifs sur la figure 4.18. Afin de simplifier l'exemple, seuls deux éléments différents seront exportés vers AMF, d'un coté le plateau, coloré en rouge, et de l'autre le reste de la vasque d'un seul bloc.

18. Frank Lloyd Wright est un célèbre architecte américain

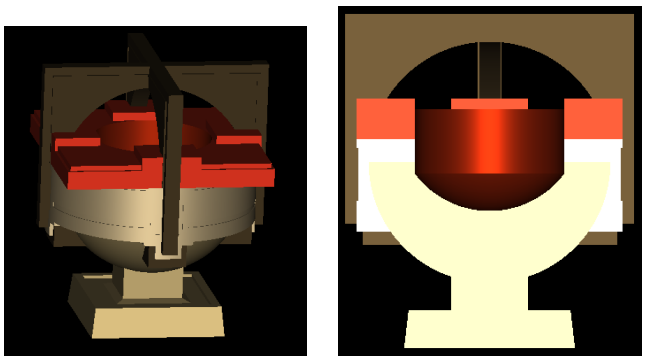

FIGURE 4.18 – Vue et coupe de la vasque de F.L Wrigth

Implémentation

L'arithmétique des formes est convertie en voxels, dont la surface est, à la suite d'un traitement de type marching Cube [19], polygonisée puis stockée en AMF. Ce traitement est effectué pour les deux sous-éléments, puis ceux-ci sont ajustés dans une constellation AMF.

Résultats

De prime abord, en regardant la vignette gauche de la figure 4.3.6, on pourrait croire que tout fonctionne. Maintenant, si l'on se réfère aux problèmes attendus évoqués en sous-section 4.3.6, il faut faire particulièrement attention aux surfaces en contact. La vignette de droite de la figure 4.3.6 identifie visuellement un effet de moirage [20], les deux surfaces se mélangeant. Il est donc visuellement évident que le support par AMF de modèles multi-pièces, issus de technologies par nature précises comme la modélisation solide, n'est à l'heure actuelle pas fonctionnel.

L'autre problème constaté, et non des moindres, lors de l'utilisation du slicer d'AMFTools, est la disparition pure et simple de certaines faces, visible sur la figure 4.20. On pourrait penser qu'il suffit d'augmenter la résolution du slicer pour que ces faces soient à nouveaux prises en compte. Or il se trouve que l'on observe le même phénomène en poussant la précision du slicer au maximum de la mémoire disponible.

19. Le marching cube est un algorithme permettant de lisser l'enveloppe d'un objet voxels, en remplaçant certains cubes voxels par des formes plus douces, comme un pan coupé par exemple.
20. Observation de perturbations issues d'erreurs de superposition

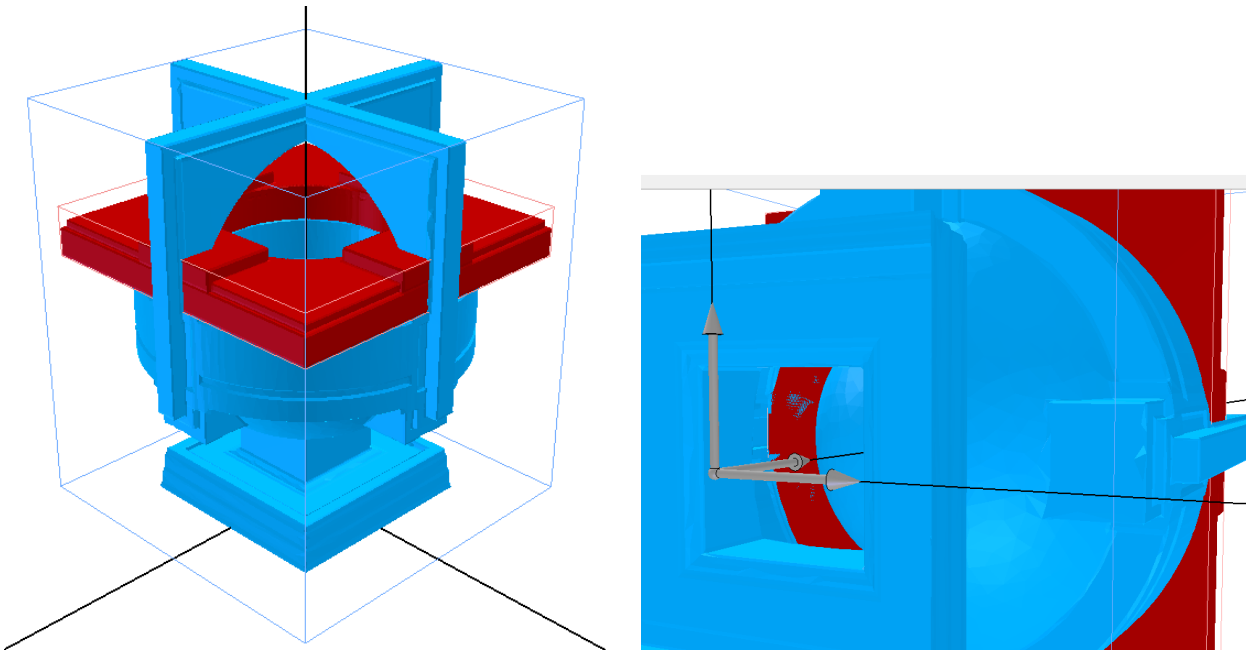

FIGURE 4.19 – Effet de moirage entre les faces

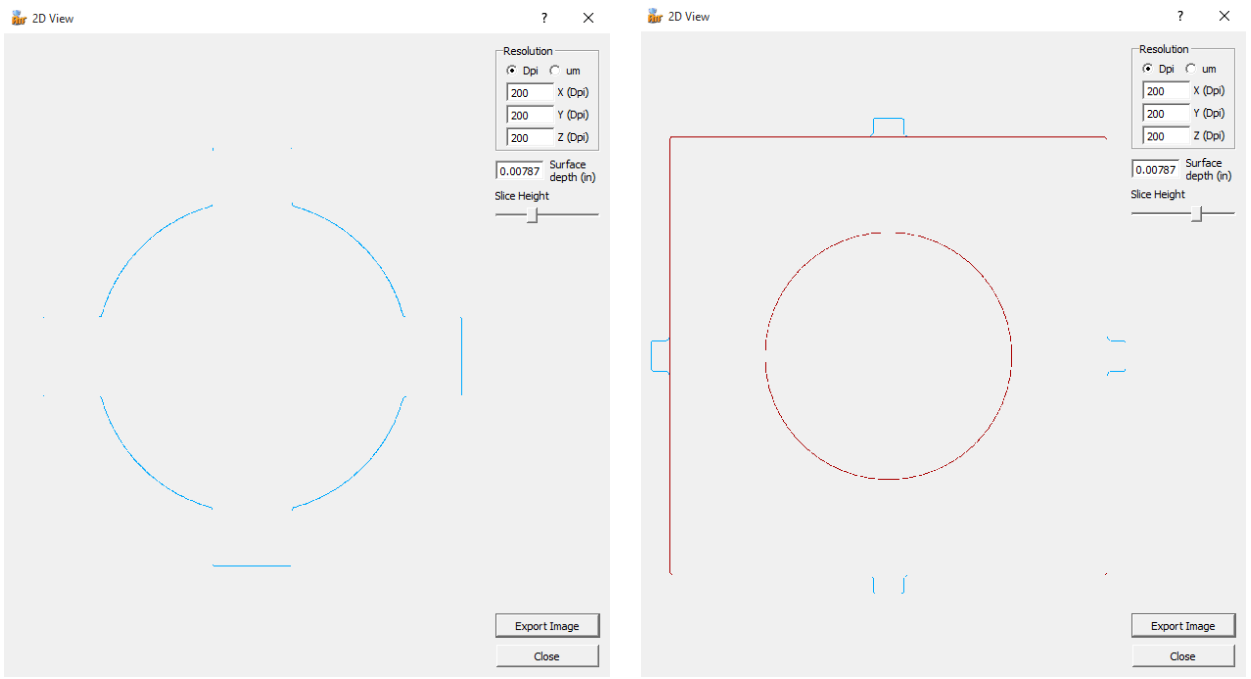

FIGURE 4.20 – Tranches générées par le slicer avec des faces manquantes

5 Conclusion

Au terme de ce stage, les objectifs de détermination des capacités du couple AMF/AMFTools sont remplis. Cela a été permis par une étude de chaque représentation à tester, au niveau théorique dans un premier temps. Puis par la détermination d'un environnement de test, déduit des observations théoriques. L'implémentation de la modélisation solide dans AMF a été déterminée après une étude de la documentation technique lorsque disponible, ou de l'analyse du code d'AMFTools lorsque la documentation faisait défaut. Enfin, l'obtention des résultats, disponibles uniquement via le slicer intégré à AMFTools a demandé une certaine abstraction. Il fallait en effet sortir ces données de leur contexte et des particularités du slicer d'AMFTools. Cela a été rendu possible par l'étude et la compréhension de l'algorithme de ce slicer.

Après la méthodologie adoptée, abordons les conclusions des tests. Il a été déterminé qu'il était possible de décrire la structure interne de l'objet via une définition fonctionnelle dans la section 4.3.1. Par la suite, une implémentation de lattice cubique a démontré la possibilité d'utiliser ces dernières dans l'état actuel des choses, abordé dans la section 4.3.3. La représentation de données voxel est également possible selon deux approches comportant chacune des avantages et des inconvénients, vus en section 4.3.4. Enfin, l'utilisation de texture pour stocker la structure interne de l'objet n'est pas fonctionnelle, tel que démontré en section 4.3.5. Un rapport d'erreur a été envoyé, mais aucune réponse n'a été proposée à cette date.

En considérant un plan plus personnel, ce stage a été réellement enrichissant. En dehors de la découverte du Canada, il m'a permis de me familiariser avec le monde de la 3D. Ce stage a également été l'occasion d'ouvrir un nouveau regard sur le monde, en découvrant l'existence de domaines d'innovations passionnants comme l'arithmétique des formes. Cet outil, par la richesse des possibilités de conception, de stockage, de traitement et de restitution des données volumiques, m'a convaincu d'y dédier de futurs travaux. Mais avant tout, cela m'a prouvé qu'il subsiste de nombreux domaines d'innovation à explorer, à inventer, à redécouvrir.

Bibliographie / Webographie

[1] 3d slash - a 3d piece of cake. https://www.3dslash.net/index.php. online : 25-08-2015. 26

[2] Amf editor. http ://amf.wikispaces.com/AMF+Editor. 11

[3] Icesl, a gpu accelerated modeler and slicer. http://www.loria.fr/~slefebvr/icesl/. Online : 25-06-2015. 14

[4] muparser - a fast math parser library. http://muparser.beltoforion.de/. Online : 25-06-2015. 12

[5] Stanley N. Burris and H.P. Sankappanavar. *A Course in Universal Algebra*. University of Waterloo, 1981. 19

[6] Thomas C Hales. The honeycomb conjecture. *Discrete & Computational Geometry*, 25(1) :1–22, 2001. 19

[7] Hod Lipson Jonathan D. Hiller. Stl 2.0 : A proposal for a universal multi-material additive manufacturing file format. *Mechanical and Aerospace Engineering Cornell Universityy*, page 13, 2009. 16, 28

[8] AMF work Group. ASTM Additive Manufacturing File Format spécification. 14, 24

Liste des illustrations

Listings

Résumé

Le monde de l'impression 3D est en pleine évolution. Dans ce contexte, plusieurs normes tentent de se démarquer par un support de plus en plus poussé des capacités actuelles et futures des imprimantes. Additive Manufacturing File Format est actuellement la principale norme internationale pour l'impression 3D, remplacant l'ancien format STL, mais de nouveaux formats émergent comme le format 3MF, poussé par un consortium industriel mené par Microsoft. Les fondateurs du projet ayant changé d'orientation, les organismes ISO et ASTM, gérants de la norme, ne sont plus au fait de ses capacités actuelles. C'est ici qu'intervient ce stage, au sein de Parrallel Geometry, experts dans le domaine de la représentation solide. Et c'est sur ces capacités d'AMF à représenter des solides que l'étude se portera. Ce rapport présente dans un premier temps le contexte ainsi que les études théoriques préliminaires effectuées. Par la suite, la méthodologie de test sera abordée, de même que les conclusions apportées par ces tests. **Mots-clés : Fabrication additive, norme, impression 3D, AMF, csg, lattice**

Abstract

The 3D printing word is undergoing a profound mutation. In such a context, several standards are trying to differentiate themselves by improving their hardware support. Additive Manufacturing file Format (AMF) is currently the main international standard for 3D printing besides the dated STL format, but there are other approaches and newcomers such as the 3MF format industrial consortium introduced by Microsoft. The support of certain parts of the AMF standard such as solid modeling has been lacking in priority compared to surfacing representations while being a part of the standard. This is the subject of this training course, within Parallel Geometry, expert in the solid modeling domain. AMF capacity to represent and communicate solid modeling will the be the main focus of the study. It will first present the context as well as prior works. In a second time, the test methodology will be described, as well as the results of said tests. **Keywords :additive manufacturing, 3D print, AMF, csg, lattice**