

# **Imagerie 3D**

## Compte rendu TP2

*Stéphane Wouters*

26 mars 2015

## Table des matières

<b>1</b>	<b>Architecture</b>	<b>3</b>
1.1	Classe Voxel . . . . .	3
1.2	Classe trianle . . . . .	3
1.3	Méthodes de la classe Voxel . . . . .	3
1.4	Alogrithme général . . . . .	3
1.5	Ecriture dans fichier . . . . .	4
<b>2</b>	<b>Résultats</b>	<b>4</b>
<b>3</b>	<b>Code source</b>	<b>4</b>
3.1	Librairie . . . . .	4
3.2	Algorithme de seuillage . . . . .	8
3.3	Fichier main . . . . .	9

# 1 Architecture

## 1.1 Classe Voxel

Un voxel est défini par son centre et par sa taille.

```
1 class Voxel {  
3     Point center;  
     float sizeX;  
5     float sizeY;  
     float sizeZ;  
7     VALUE value;  
  
9     Voxel(float x, float y, float z, VALUE value) {  
         this->value = value;  
11        center.set(x,y,z);  
    }  
13  
    ...
```

## 1.2 Classe trianle

Un voxel est défini par 3 coordonnées

```
class Triangle {  
2  
    Point p1;  
4    Point p2;  
    Point p3;  
6    ...
```

## 1.3 Méthodes de la classe Voxel

A partir d'un voxel, les méthodes suivantes sont définies :

- **Point\* getSommets()** Retourne la position des 8 sommets du voxel
- **Point\* getAdj()** Retourne la position des 6 voxels adjacent (les centres)
- **Point\* getTriangles(int numeroFace)** Retourne les deux triangle de la face numéro N (de 0 à 6)

## 1.4 Alogrithme général

La fonction retourne la liste des triangles générés.

```
vector<Triangle> seuillage(Image img, int seuil) {  
2  
    vector<Triangle> out;  
4  
    // Parcours de l'image  
6    for (int i = 0; i < img.sizeX; ++i) {  
        for (int j = 0; j < img.sizeY; ++j) {  
8            for (int k = 0; k < img.sizeZ; ++k) {  
  
10                // Calcul du voxel  
                VALUE value = img.getValue(i,j,k);
```

```

12     Voxel v(i,j,k, value);
14     if (v.getValue() > seuil) {
16         // Parcours des voisins
17         for (auto adj : v.getAdj()) {
18             if (img.getValue(adj.x,adj.y,adj.z) < seuil) {
19
20                 // Calcul des triangles
21                 Triangle* triangles = v.getTriangles(face);
22                 out.push_back(triangles[0]);
23                 out.push_back(triangles[1]);
24             }
25         }
26     }
27 }
28 }
29 }
30 return out;
31 }

```

## 1.5 Ecriture dans fichier

Pour retranscrire les triangles dans un fichier, on ajoute une méthode toString dans Triangle :

```

string toString() {
2     string s = "facet normal 0 0 0\n";
3     s += "outer loop\n";
4     s += "vertex "+p1.toString()+"\n";
5     s += "vertex "+p2.toString()+"\n";
6     s += "vertex "+p3.toString()+"\n";
7     s += "endloop\nendfacet";
8     return s;
9 }

```

On affiche toString() sur chacun des triangles de sortie, et on redirige la sortie standard vers un fichier STL à l'exécution du fichier binaire..

## 2 Résultats

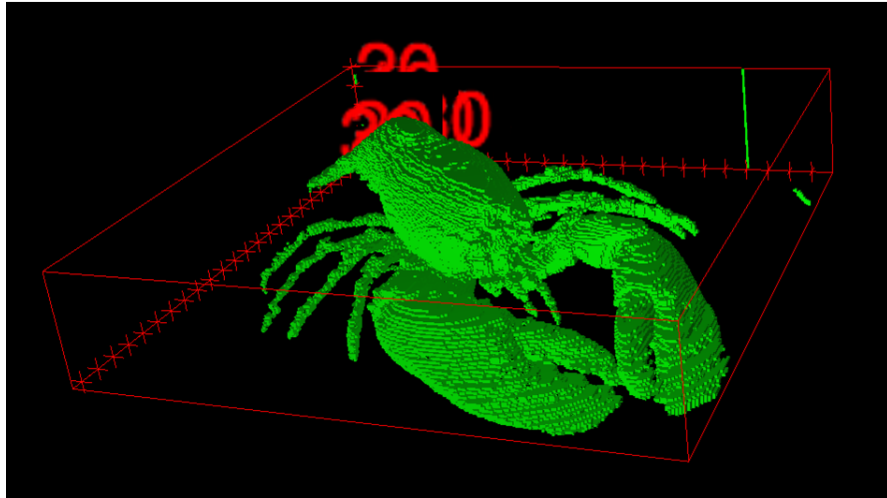


FIGURE 1 – Whatisit sous FIGI

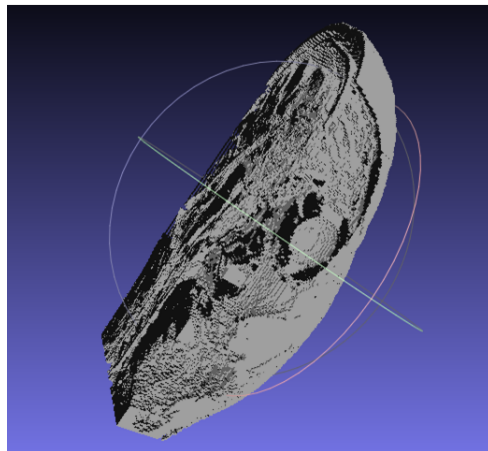


FIGURE 2 – Brainix sous MeshLab

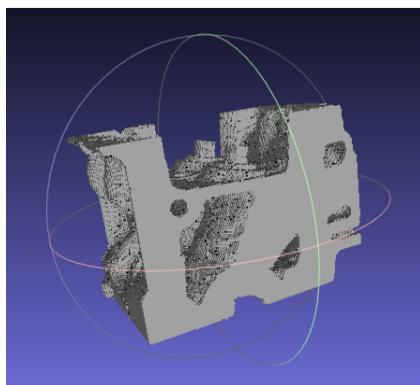


FIGURE 3 – Engine, seuil 100 sous MeshLab

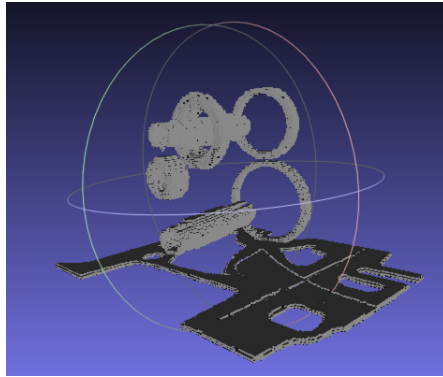


FIGURE 4 – Engine, seuil 200 sous MeshLab

## 3 Code source

### 3.1 Librairie

```

1  #ifndef VOXEL_PPM
   #define VOXEL_PPM
3
   #include <iostream>
5  #include <stdlib.h>
   #include <stdio.h>
7  #include <string>
   #include <math.h>
9  #include "Image.h"

11 typedef unsigned short VALUE;

13 using namespace std;

15 class Point {
   public:
17     float x;
     float y;
19     float z;

21     Point() {
         set(0,0,0);
23     }

25     Point(float x, float y, float z) {
         set(x,y,z);
27     }

29     void set(float x, float y, float z) {
         this->x = x;
31         this->y = y;
         this->z = z;
33     }

35     string toString() {
         char s[250];
37         sprintf(s, "%f %f %f", x, y, z);

```

```

    return s;
39 }
};
41
class Triangle {
43 public:
    Point p1;
45     Point p2;
        Point p3;
47
    string toString() {
49         string s = "facet normal 0 0 0\n";
            s += "outer loop\n";
51         s += "vertex "+p1.toString() +"\n";
            s += "vertex "+p2.toString() +"\n";
53         s += "vertex "+p3.toString() +"\n";
            s += "endloop\nendfacet";
55         return s;
        }
57
};
59
61 class Voxel {
    public:
63
        Point center;
65         float sizeX;
            float sizeY;
67             float sizeZ;
                VALUE value;
69
        Voxel(float x, float y, float z, VALUE value) {
71             this->value = value;
                center.set(x,y,z);
73                 sizeX = 1;
                    sizeY = 1;
75                     sizeZ = 1;
        }
77
        Point* getSommets() {
79             Point* sommets = new Point[8];
                sommets[0].set(center.x -sizeX/2, center.y -sizeY/2, center.z -sizeZ/2)
                    ;
81                 sommets[1].set(center.x +sizeX/2, center.y -sizeY/2, center.z -sizeZ/2)
                    ;
                        sommets[2].set(center.x +sizeX/2, center.y +sizeY/2, center.z -sizeZ/2)
                            ;
83                             sommets[3].set(center.x -sizeX/2, center.y +sizeY/2, center.z -sizeZ/2)
                                    ;
                                        sommets[4].set(center.x -sizeX/2, center.y -sizeY/2, center.z +sizeZ/2)
                                            ;
85                                                sommets[5].set(center.x +sizeX/2, center.y -sizeY/2, center.z +sizeZ/2)
                                                    ;
                                                        sommets[6].set(center.x +sizeX/2, center.y +sizeY/2, center.z +sizeZ/2)
                                                            ;
87                                                                sommets[7].set(center.x -sizeX/2, center.y +sizeY/2, center.z +sizeZ/2)
                                                                    ;
                                                                        return sommets;
89     }
}

```

```

91 // Numéro de la face de 0 à 5
Triangle* getTriangles(int nFace) {
93     Triangle* triangles = new Triangle[2];
        Point* sommets = getSommets();
95
        switch (nFace) {
97             case 0: // Droite
                triangles[0].p1 = sommets[1];
99                 triangles[0].p2 = sommets[2];
                    triangles[0].p3 = sommets[5];
101                 triangles[1].p1 = sommets[2];
                    triangles[1].p2 = sommets[6];
103                 triangles[1].p3 = sommets[5];
                    break;
105             case 1: // Gauche
                triangles[0].p1 = sommets[4];
107                 triangles[0].p2 = sommets[3];
                    triangles[0].p3 = sommets[0];
109                 triangles[1].p1 = sommets[4];
                    triangles[1].p2 = sommets[6];
111                 triangles[1].p3 = sommets[3];
                    break;
113             case 2: // Derrière
                triangles[0].p1 = sommets[6];
115                 triangles[0].p2 = sommets[2];
                    triangles[0].p3 = sommets[3];
117                 triangles[1].p1 = sommets[6];
                    triangles[1].p2 = sommets[3];
119                 triangles[1].p3 = sommets[7];
                    break;
121             case 3: // Devant
                triangles[0].p1 = sommets[0];
123                 triangles[0].p2 = sommets[1];
                    triangles[0].p3 = sommets[5];
125                 triangles[1].p1 = sommets[0];
                    triangles[1].p2 = sommets[5];
127                 triangles[1].p3 = sommets[4];
                    break;
129             case 4: // Haut
                triangles[0].p1 = sommets[4];
131                 triangles[0].p2 = sommets[5];
                    triangles[0].p3 = sommets[6];
133                 triangles[1].p1 = sommets[6];
                    triangles[1].p2 = sommets[7];
135                 triangles[1].p3 = sommets[4];
                    break;
137             case 5: // Bas
                triangles[0].p1 = sommets[3];
139                 triangles[0].p2 = sommets[2];
                    triangles[0].p3 = sommets[1];
141                 triangles[1].p1 = sommets[3];
                    triangles[1].p2 = sommets[1];
143                 triangles[1].p3 = sommets[0];
                    break;
145         }

147     return triangles;
}
149
Point* getAdj() {
151     Point* voxels = new Point[6];

```



```

153     voxels[0].set(center.x + 1, center.y, center.z);
    voxels[1].set(center.x - 1, center.y, center.z);
    voxels[2].set(center.x, center.y + 1, center.z);
155     voxels[3].set(center.x, center.y - 1, center.z);
    voxels[4].set(center.x, center.y, center.z + 1);
157     voxels[5].set(center.x, center.y, center.z - 1);
    return voxels;
159 }

161 };

163
#endif

```

### 3.2 Algorithme de seuillage

```

#ifndef IMAGETOOLS_PPM
2 #define IMAGETOOLS_PPM

4 #include <iostream>
#include <stdlib.h>
6 #include <stdio.h>
#include <string.h>
8 #include <math.h>
#include "Image.h"
10 #include "Voxel.h"
#include <Vector>

12
vector<Triangle> seuillage(Image img, int seuil) {
14
    vector<Triangle> out;

16
    for (int i = 0; i < img.sizeX; ++i) {
18         for (int j = 0; j < img.sizeY; ++j) {
                for (int k = 0; k < img.sizeZ; ++k) {
20                     VALUE value = img.getValue(i,j,k);
                     Voxel v(i,j,k, value);
22                     if (value > seuil) {
                         Point* adjs = v.getAdj();
24                         for (int face = 0; face < 6; ++face) {
                             Point adj = adjs[face];
26                             VALUE valueADJ = img.getValue(adj.x,adj.y,adj.z);
                             if (valueADJ < seuil) {
28                                 Triangle* triangles = v.getTriangles(face);
                                 out.push_back(triangles[0]);
30                                 out.push_back(triangles[1]);
                            }
32                     }
                }
            }
34        }
    }
36    }
    return out;
38 }

40 void print(vector<Triangle> tab) {
    for (auto var : tab) {
42         cout << var.toString() << endl;
    }
}

```

```
44 }  
#endif
```

### 3.3 Fichier main

```
1 #include "../lib/Image.h"  
#include "../lib/Voxel.h"  
3 #include "../lib/ImageTools.h"  
#include <iostream>  
5 #include <algorithm>  
  
7 using namespace std;  
  
9 void printModel(const char* path, int seuil, int sizeX, int sizeY, int  
    sizeZ) {  
    Image in(sizeX, sizeY, sizeZ);  
11    in.load(path);  
    cout << "solid name" << endl;  
13    vector<Triangle> triangles = seuillage(in, seuil);  
    print(triangles);  
15    cout << "endsolid name" << endl;  
    }  
17  
int main() {  
19    //printModel("../ressources/BRAINIX/brainix.256x256x100.0.9375x0.9375x1  
        .5.img", 200, 256, 256, 100);  
    printModel("../ressources/MANIX/manixSansIV.512x512x48.0.4570x0.4570x3.0.  
        img", 1250, 512, 512, 48);  
21    //printModel("../ressources/BEAUFIX/beaufix.448x576x72.0.6250x0.6250x1.4.  
        img", 120, 576, 72, 448);  
    //printModel("../ressources/WHATISIT/whatisit.301x324x56.1.1.1.4.img",  
        50, 301, 324, 56);  
23    //printModel("../ressources/engine/engine.256x256x128.1x1x1.img", 200,  
        256, 256, 128);  
    }
```