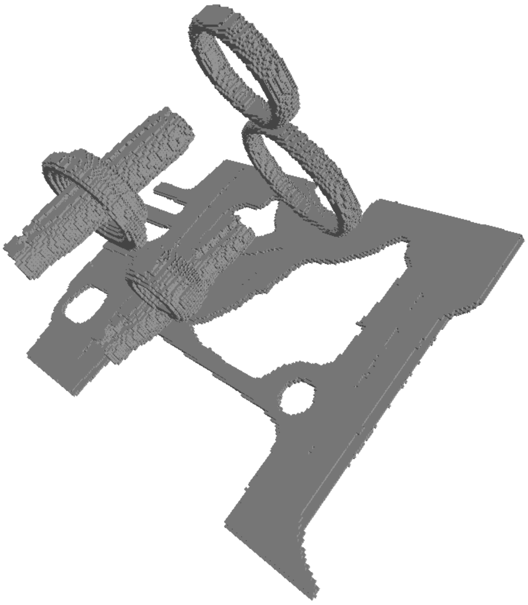
Compte rendu TP image 3D-2
Traitement de l'image

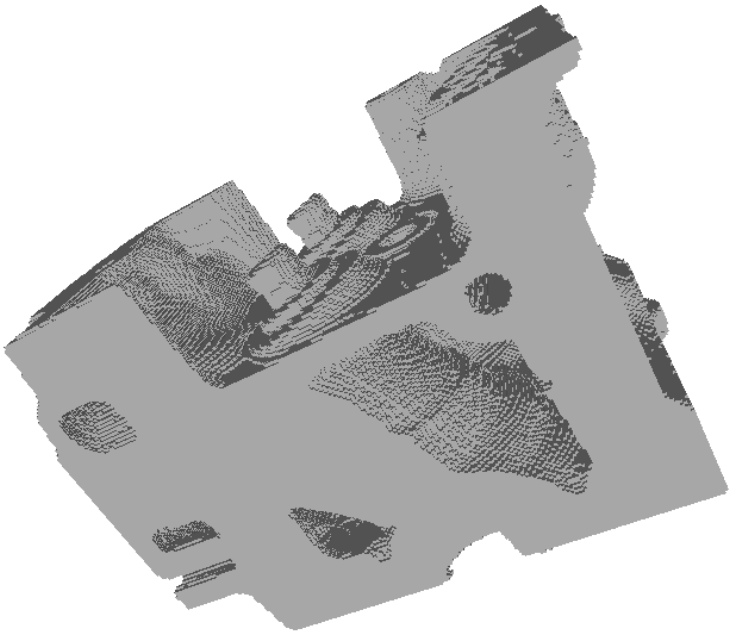
Tristan Cossin
Master IMAGINA

MAI 2017

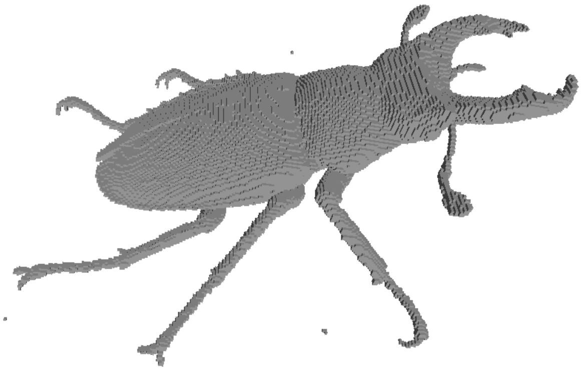
Engine (seuil = 200):



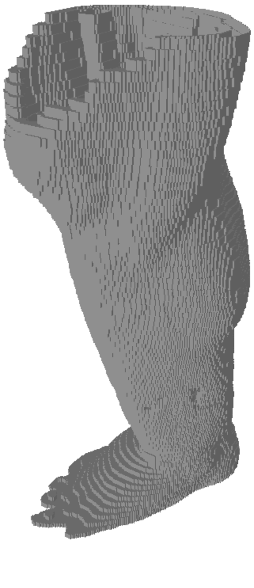
Engine (seuil = 100):



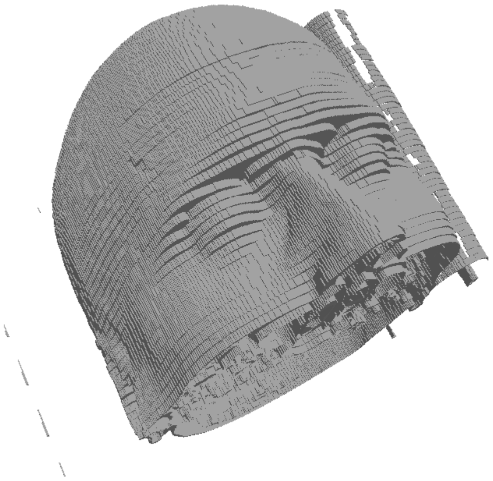
Whatisit (seuil = 100):



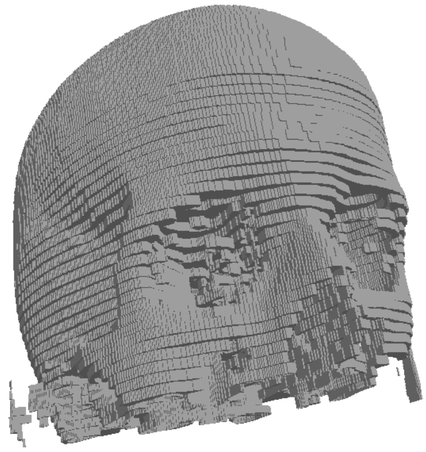
StatueLeg (seuil = 50):



Manix (seuil = 900):



Manix (seuil = 1100):



Méthode d’enregistrement des faces:

```
void ToString(ofstream &file, face &t)
{
	file << "facet normal 0 0 0" << endl;
	file << "outer loop"<< endl;
	file << "vertex " << t.v1.x << " " << t.v1.y << " " << t.v1.z << endl;
	file << "vertex " << t.v2.x << " " << t.v2.y << " " << t.v2.z << endl;
	file << "vertex " << t.v3.x << " " << t.v3.y << " " << t.v3.z << endl;
	file << "endloop " << endl;
	file << "endfacet " << endl;
}
```

Méthode getValue:

```
unsigned short getValue(unsigned short *buffer, int x, int y, int z)
{
	return inverserOctet(buffer[(z * tailleX * tailleY) + ((tailleY - y - 1) * tailleX) + x ]);
}
```

Méthode MarchingCube:

```
void MarchingCube(unsigned short *buffer, int threashold)
{
	std::ofstream file;
	file.open("out.stl");

	file << "solid test" << endl;

	//pour tous les voxels
	for(int i = 1; i < tailleX - 1; i++)
	{
		for(int j = 1; j < tailleY - 1; j++)
		{
			for(int k = 1; k < tailleZ - 1; k++)
			{
				//si le voxel est supérieur ou égal au seuil
				if(getValue(buffer, i, j, k) >= threashold)
				{
					//tableau des sommets du voxel
					vector<point> vertices;

					point v1;
					v1.x = (i - 0.5) * sizeX;
					v1.y = (j - 0.5) * sizeY;
					v1.z = (k - 0.5) * sizeZ;
					vertices.push_back(v1);

					point v2;
					v2.x = (i + 0.5) * sizeX;
					v2.y = (j - 0.5) * sizeY;
					v2.z = (k - 0.5) * sizeZ;
					vertices.push_back(v2);

					point v3;
					v3.x = (i + 0.5) * sizeX;
					v3.y = (j + 0.5) * sizeY;
					v3.z = (k - 0.5) * sizeZ;
					vertices.push_back(v3);
```

```
point v4;
v4.x = (i - 0.5) * sizeX;
v4.y = (j + 0.5) * sizeY;
v4.z = (k - 0.5) * sizeZ;
vertices.push_back(v4);

point v5;
v5.x = (i - 0.5) * sizeX;
v5.y = (j - 0.5) * sizeY;
v5.z = (k + 0.5) * sizeZ;
vertices.push_back(v5);

point v6;
v6.x = (i + 0.5) * sizeX;
v6.y = (j - 0.5) * sizeY;
v6.z = (k + 0.5) * sizeZ;
vertices.push_back(v6);

point v7;
v7.x = (i + 0.5) * sizeX;
v7.y = (j + 0.5) * sizeY;
v7.z = (k + 0.5) * sizeZ;
vertices.push_back(v7);

point v8;
v8.x = (i - 0.5) * sizeX;
v8.y = (j + 0.5) * sizeY;
v8.z = (k + 0.5) * sizeZ;
vertices.push_back(v8);

//on check les 6 voxels adjacent
if(getValue(buffer, i - 1, j, k) < threashold)
{
        face t1;
        t1.v1 = vertices[0];
        t1.v2 = vertices[3];
        t1.v3 = vertices[7];

        face t2;
        t2.v1 = vertices[7];
        t2.v2 = vertices[4];
        t2.v3 = vertices[0];

        ToString(file, t1);
        ToString(file, t2);
}

if(getValue(buffer, i + 1, j, k) < threashold)
{
        face t1;
        t1.v1 = vertices[2];
        t1.v2 = vertices[6];
        t1.v3 = vertices[5];

        face t2;
        t2.v1 = vertices[5];
        t2.v2 = vertices[1];
        t2.v3 = vertices[2];

        ToString(file, t1);
        ToString(file, t2);
}
```

```
if(getValue(buffer, i, j - 1, k) < threashold)
{
        face t1;
        t1.v1 = vertices[0];
        t1.v2 = vertices[1];
        t1.v3 = vertices[5];

        face t2;
        t2.v1 = vertices[5];
        t2.v2 = vertices[4];
        t2.v3 = vertices[0];

        ToString(file, t1);
        ToString(file, t2);
}

if(getValue(buffer, i, j + 1, k) < threashold)
{
        face t1;
        t1.v1 = vertices[7];
        t1.v2 = vertices[6];
        t1.v3 = vertices[2];

        face t2;
        t2.v1 = vertices[2];
        t2.v2 = vertices[3];
        t2.v3 = vertices[0];

        ToString(file, t1);
        ToString(file, t2);
}

if(getValue(buffer, i, j, k - 1) < threashold)
{
        face t1;
        t1.v1 = vertices[0];
        t1.v2 = vertices[1];
        t1.v3 = vertices[2];

        face t2;
        t2.v1 = vertices[2];
        t2.v2 = vertices[3];
        t2.v3 = vertices[0];

        ToString(file, t1);
        ToString(file, t2);
}

if(getValue(buffer, i, j, k + 1) < threashold)
{
        face t1;
        t1.v1 = vertices[4];
        t1.v2 = vertices[5];
        t1.v3 = vertices[6];

        face t2;
        t2.v1 = vertices[6];
        t2.v2 = vertices[7];
        t2.v3 = vertices[4];

        ToString(file, t1);
        ToString(file, t2);
}
```

```
                        }
                    }
                }
            }

            file << "endsolid test";

            file.close();
}
```