Modélisation et géométrie discrète

Compte-rendu TP8

Représentation volumique

Louis Jean
Master 1 IMAGINE
Université de Montpellier

21 novembre 2023

Table des matières

1 Introduction

Dans ce TP, nous explorons le concept de représentation volumique, un élément fondamental dans le domaine de la modélisation géométrique et de la visualisation 3D. L'objectif principal est de développer une compréhension de base des techniques permettant de représenter et de manipuler des formes en trois dimensions à l'aide de voxels, qui sont des unités volumétriques analogues aux pixels en 2D.

2 Affichage d'un voxel

La première étape était de créer "l'atome" de ce TP : le voxel. J'ai choisi de faire une struct et de décrire un simple cube.

```
1 struct Voxel {
2     Vec3 center;
3     double side;
4 };
```

Figure 1: Définition d'un voxel

J'ai ensuite déduit les points d'un voxel quelconque.

```
void getVoxelPoints(Voxel v, Vec3* voxelPoints) {
    voxelPoints[0] = Vec3(v.center[0] - 0.5*v.side,v.
        ↪ center[1] - 0.5*v.side,v.center[2] - 0.5*v.side
        ↪ ); // En bas à gauche au premier plan
    voxelPoints[1] = Vec3(v.center[0] + 0.5*v.side,v.
        ↪ center[1] - 0.5*v.side,v.center[2] - 0.5*v.side
        ↪ ); // En bas à droite au premier plan
    voxelPoints[2] = Vec3(v.center[0] + 0.5*v.side,v.
        ↪ center[1] + 0.5*v.side,v.center[2] - 0.5*v.side
        ↪ ); // En haut à droite au premier plan
    voxelPoints[3] = Vec3(v.center[0] - 0.5*v.side,v.
        ↪ center[1] + 0.5*v.side,v.center[2] - 0.5*v.side
        ↪ ); // En haut à gauche au premier plan
    voxelPoints[4] = Vec3(v.center[0] - 0.5*v.side,v.
        ↪ center[1] - 0.5*v.side,v.center[2] + 0.5*v.side
        ↪ ); // En bas à gauche au fond
    voxelPoints[5] = Vec3(v.center[0] + 0.5*v.side,v.
        ↪ center[1] - 0.5*v.side,v.center[2] + 0.5*v.side
        ↪ ); // En bas à droite au fond
    voxelPoints[6] = Vec3(v.center[0] + 0.5*v.side,v.
        ↪ center[1] + 0.5*v.side,v.center[2] + 0.5*v.side
        ↪ ); // En haut à droite au fond
    voxelPoints[7] = Vec3(v.center[0] - 0.5*v.side,v.
        ↪ center[1] + 0.5*v.side,v.center[2] + 0.5*v.side
        ↪ ); // En haut à gauche au fond
}
```

Figure 2: Déduction des points du voxel

Puis, j'ai relié les triangles de chaque face du voxel (le code est trop long pour être affiché ici), et j'ai enfin pu afficher mon premier voxel.

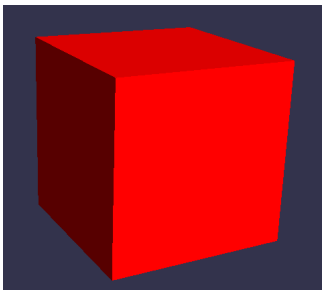


Figure 3: Rendu d'un simple voxel

3 Affichage d'une grille de voxels

J'ai ensuite décidé d'afficher une grille de voxels, puisque tout le TP se base là-dessus.

```
1  void drawGrid(Vec3 origin, double resolution) {
2      Voxel v;
3      v.side = 1/resolution;
4      for(int i = 0; i < resolution; i++) {
5          for(int j = 0; j < resolution; j++) {
6              for(int k = 0; k < resolution; k++) {
7                  v.center = Vec3(i * v.side, j * v.side, k
                       ↪  * v.side);
8                  drawVoxel(v.center,v.side);
9              }
10         }
11     }
12 }
```

Figure 4: Création d'une grille de voxels

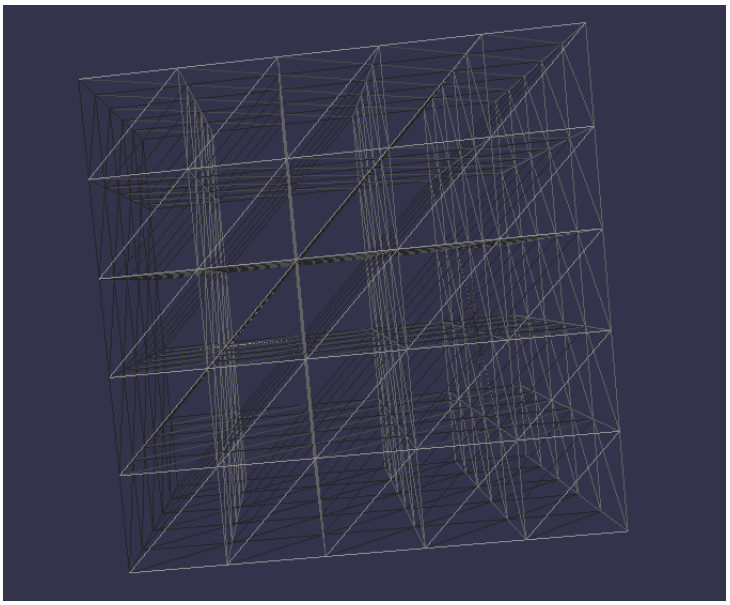


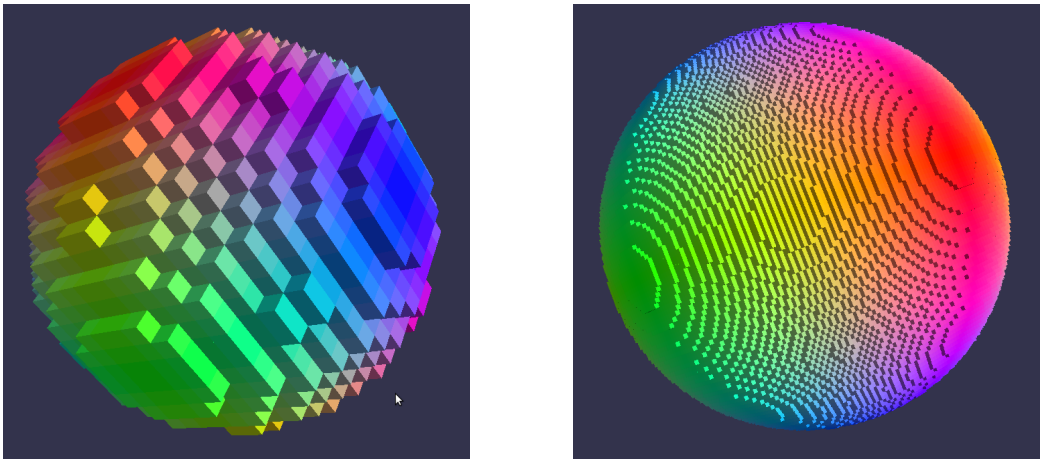
Figure 5: Affichage d'une grille de voxels 5x5x5

4 Première forme volumétrique : la sphère

Pour dessiner une sphère à l'aide de voxels, j'ai créé une grille de voxels, et pour chaque voxel de cette grille, j'ai testé si ce dernier appartenait à la sphère. Plus l'on augmente resolution, plus la sphère sera composé de voxels, et donc détaillée.

```
1  void drawSphereVolumic(Vec3 center, double rayon, double
       ↪ resolution) {
2      Voxel v;
3      v.side = 2 * rayon / resolution;
4      for(int i = 0; i < resolution; i++) {
5          for(int j = 0; j < resolution; j++) {
6              for(int k = 0; k < resolution; k++) {
7                  v.center = Vec3(center[0] - rayon + i * v
                       ↪ .side + v.side / 2,
8                  center[1] - rayon + j * v.side + v.side /
                       ↪  2,
9                  center[2] - rayon + k * v.side + v.side /
                       ↪  2);
10                 if((center - v.center).length() <= rayon)
                       ↪  {
11                     drawVoxel(v.center, v.side);
12                 }
13             }
14         }
15     }
16 }
```

Figure 6: Code pour créer une sphère de voxels



(a) resolution = 20 (b) resolution = 100

Figure 7: Rendus de 2 sphères avec les voxels

5 Seconde représentation volumique : le cylindre

Pour cette deuxième figure, le principe reste le même, mais les conditions d'appartenance changent, avec notamment une vérification à faire autour de l'axe du cylindre.

```
1  void drawCylinderVolumic(Vec3 axisOrigin, Vec3 axisVector
       ↪ , double rayon, double resolution) {
2      Voxel v;
3      v.side = 2 * rayon / resolution;
4      double cylinderHeight = axisVector.length();
5      for(int i = 0; i < resolution; i++) {
6          for(int j = 0; j < resolution; j++) {
7              for(int k = 0; k < resolution; k++) {
8                  v.center = Vec3(axisOrigin[0] - rayon + i
                       ↪  * v.side + v.side / 2,
9                                  axisOrigin[1] - rayon + j
                                       ↪  * v.side + v.side
                                       ↪ / 2,
10                                 axisOrigin[2] + k * v.
                                       ↪ side + v.side / 2);
11                 double dx = v.center[0] - axisOrigin[0];
12                 double dy = v.center[1] - axisOrigin[1];
13                 if(dx * dx + dy * dy <= rayon * rayon) {
14                     if(v.center[2] >= axisOrigin[2] && v.
                           ↪ center[2] <= axisOrigin[2] +
                           ↪ cylinderHeight) {
15                         //std::cout<<v.center[2]<<std::
                               ↪ endl;
16                         drawVoxel(v.center,v.side);
17                     }
18                 }
19             }
20         }
21     }
22 }
```

Figure 8: Code pour créer un cylindre de voxels

6 Intersection, union et soustraction d'un cylindre et d'une sphère

Dans ce dernier exercice, l'idée était de combiner les deux volumes de plusieurs manières afin d'obtenir des rendus intéressants. Globalement, les codes des trois fonctions sont assez similaires, et se contentent de faire des combinaisons de conditions.

6.1 Intersection

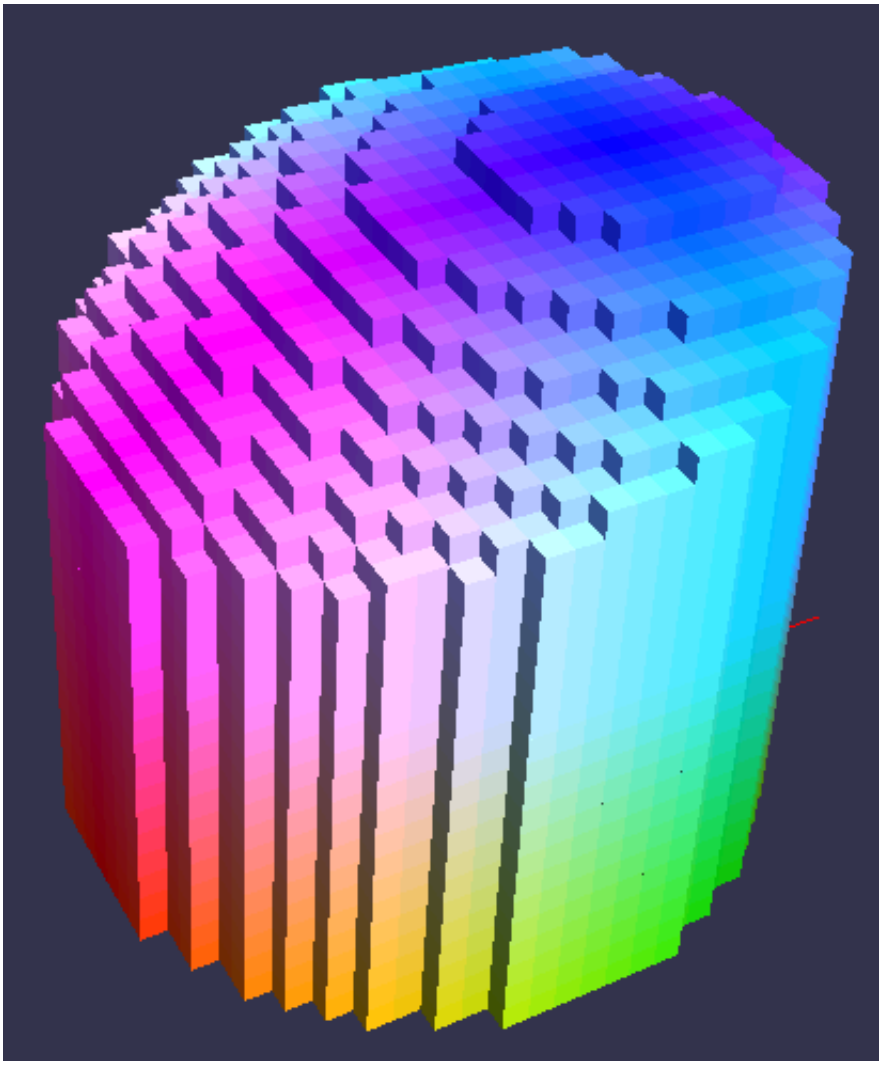


Figure 9: Intersection d'une sphère et d'un cylindre

6.2 Union

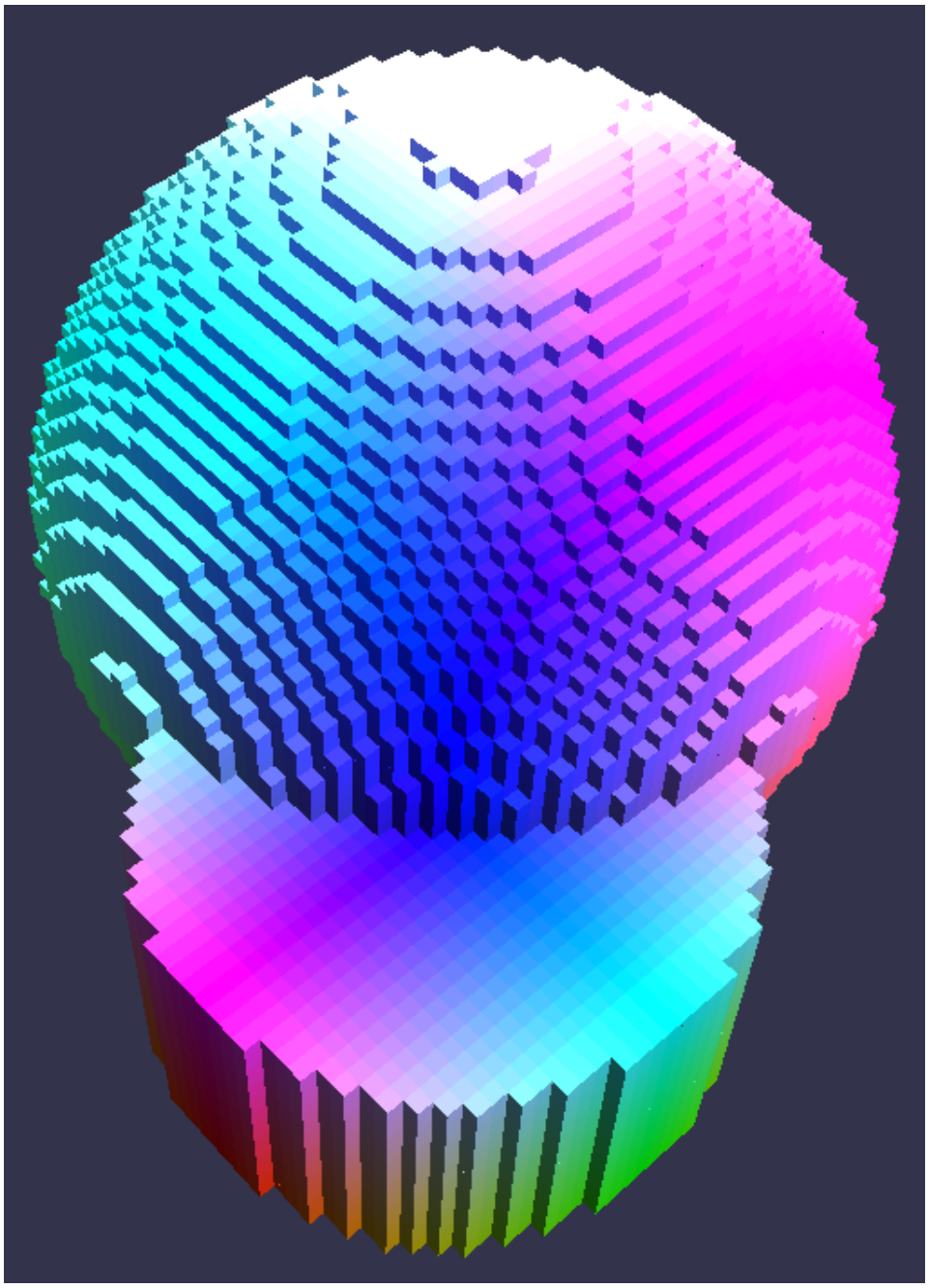


Figure 10: Union d'une sphère et d'un cylindre

6.3 Soustraction

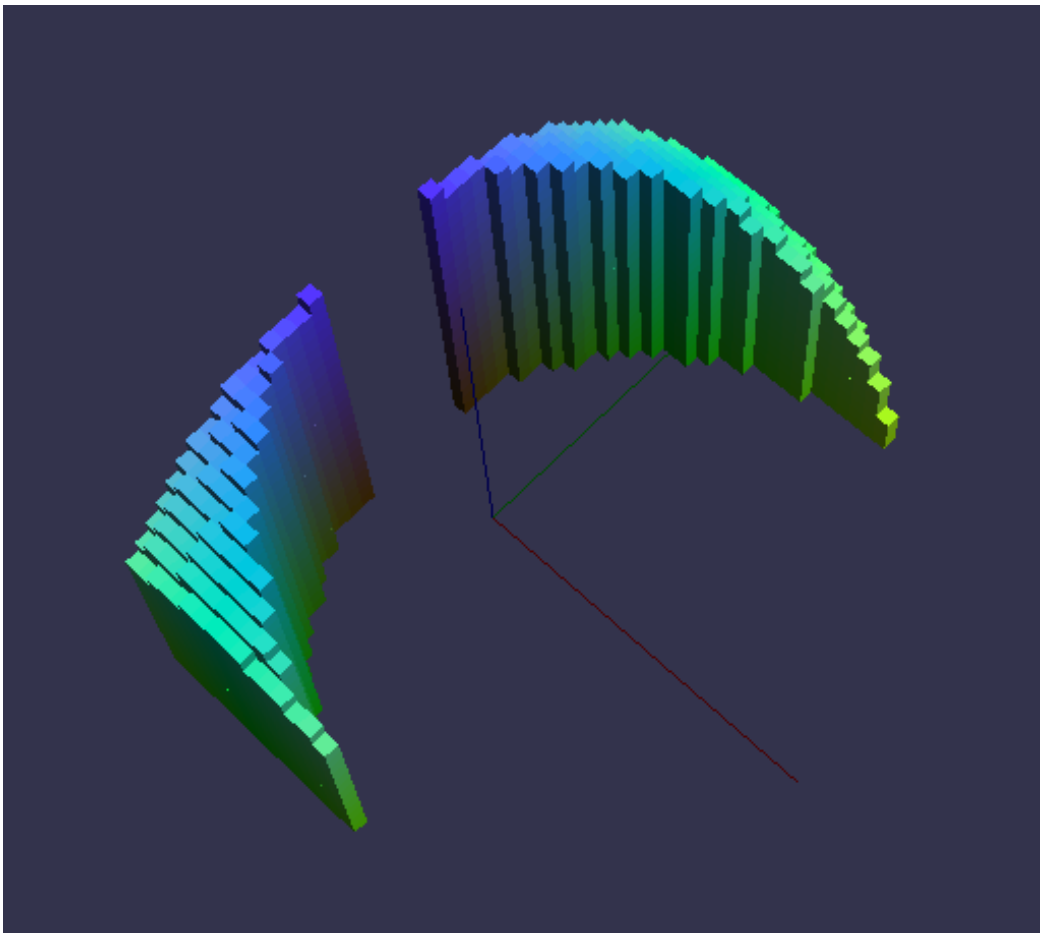


Figure 11: Soustraction d'un cylindre à une sphère

Merci pour le temps et l'attention que vous avez consacrés à la lecture de ce compte-rendu.