Rapport Développement Logiciel

Badr YOUBI IDRISSI

1er mai 2018

1 Introduction et fonctionnement général

Dans ce projet nous codons un programme pour visualiser les isolignes d'un signal émit de plusieurs tétraèdres.

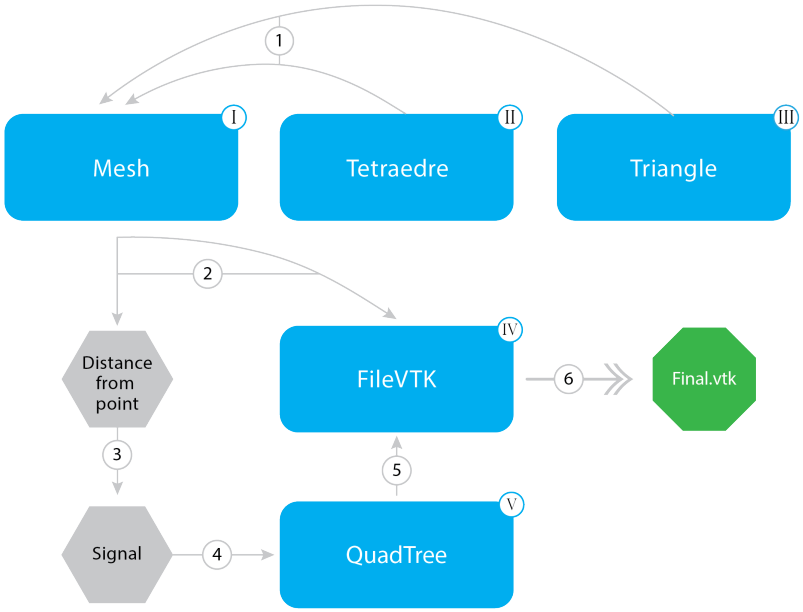

FIGURE 1 – Schéma du fonctionnement général

On commence par créer l'objet 'Mesh' (I) qui correspond à l'ensemble des pods émétant le signal. L'objet mesh a besoin des points correspondant à chaque pod qui est un 'Tétraèdre' (II) qui lui même est un ensemble de 'Triangles' (III) (Étape 1). Cet objet Mesh a la méthode distance from point qui renvoie la liste des distances entre un point et les pods (Étape 2). Ces distances servent ensuite à calculer le signal en tout point de l'espace (Étape 3). on crée ensuite un plan à la hauteur 0.01 dont on va rafiner le maillage avec l'objet 'QuadTree' (V) (Étape 4). Ensuite l'ensemble des points et les valeurs du signal en ces points est passé à l'objet 'FileVTK' (IV) (Étape 5) qui va écrire le fichier "Final.vtk" (Étape 6).

2 Description du code

2.1 Objet III : Triangle

```
class tri:
        def __init__(self, point1, point2, point3):
                ...
        def distance_to_a_point(self, P):
                ...
```

L'objet triangle sert à stocker les points du triangle et à calculer la distance entre un point et à calculer la distance entre un point quelconque et le triangle.

```
def __init__(self, point1, point2, point3):
        self.points = [np.array(point1), np.array(point2), np.array(point3)]
        p1p2 = self.points[1]-self.points[0]
        p1p3 = self.points[2]-self.points[0]
        self.normal = np.cross(p1p2, p1p3)
        self.normal = self.normal/np.linalg.norm(self.normal)
```

On enregistre les points A, B et C et on calcule la normale $\overrightarrow{n'} = \overrightarrow{AB} \times \overrightarrow{AC}$ puis $\vec{n} = \frac{\overrightarrow{n'}}{\|\overrightarrow{n'}\|}$

Pour la fonction de calcul de distance la fonction projète le point sur le plab du triangle puis selon les valeurs des coordonnées barycentriques on retourne la plus petite distance à un coin ou à un bord. Au vu du grand nombre de cas, j'ai opté pour reprendre un code déjà fait qui applique ce principe (voir code) en l'adaptant en rajoutant la condition

```
normal_side  = np.dot(self.normal, P) > np.dot(self.normal,self.points[0])
```

car un point est du bon coté du plan du triangle si $\langle \vec{p}, \vec{n} \rangle > \langle \vec{A}, \vec{n} \rangle$

2.2 Objet II : Tetra

```
class tetra:
        def __init__(self, point1, point2, point3, point4):
                ...
        def distance_to_a_point(self, p):
                ...
```

L'objet tetra est un ensemble de triangles correctement orientés (normales vers l'extérieur) qui peut calculer la distance entre un point et le tetra.

```
def __init__(self, point1, point2, point3, point4):
        self.points = [point1, point2,point3,point4]
        self.faces = [tri(point1,point3,point2), tri(point1,point4,point3),
                                tri(point2,point3,point4), tri(point1,point2,point4)]
```

L'ordre des points ici est important pour bien orienter les normales.

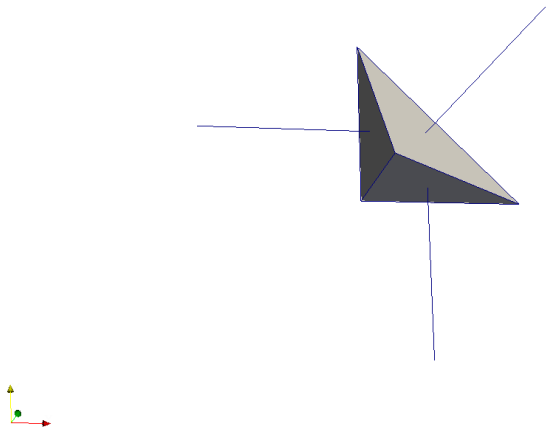

FIGURE 2 – Tétraèdre avec normales

```
def distance_to_a_point(self, p):
        d_min = np.inf
        outside = False

        for f in self.faces:
                [normal_side, d_minTri] = f.distance_to_a_point(p)
                outside = outside or normal_side
                if d_minTri < d_min:
                        d_min = d_minTri

        return [outside, d_min]
```

la distance à un Tetraèdre est le minimum de distance aux autres triangles. sachant que si le point est du mauvais côté la distance est infini.

2.3 Objet I : mesh

```
class tetra:
        def __init__ (self, DefenseFile):
                ...
        def getNextValue(self)::
                ...
        def toVector(self,string):
                ...
        def Create_Mesh_And_Connectivity_List(self, List_Of_Point, Connectivity):
                ...
        def Distance_Between_a_Point_and_the_Modules(self, p):
                ...
```