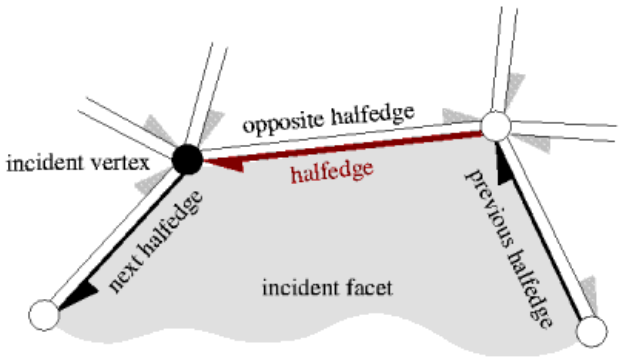


RAPPORT TP SEGMENTATION

Réalisé par :
TABAA Abdelkader / M1 Informatique

UNIVERSITÉ Clermont Auvergne

ISIMA

1- Calcule d’une propriété locale :

```python
def angle_ddrl(mesh):
    angle=[]
    for face in mesh.facets:
        angles_facet=[]
        edge_facet=face.halfedge
        suivante_halfedge=edge_facet.next
        angle_facet=edge_facet.get_angle_normal()
        angles_facet.append(angle_facet)
        while edge_facet.index != suivante_halfedge.index:
            angle_facet=suivante_halfedge.get_angle_normal()
            angles_facet.append(angle_facet)
            suivante_halfedge=suivante_halfedge.next
        angle.append(angles_facet)
    return angle
```

Au début j’ai commencé par faire une exploration selon les faces (Mesh.facets) à chaque fois je récupère la demi-arrêt du face et je récupère aussi la demi arrête suivante et puis je prends l’angle diédrale de cette demi-arrêt et je l’ai mis dans le tableau associé aux angles des facets, après tant que l’indice de la demi-arrêt (la demi-arrêt de la première facet) est diffèrent de l’indice de la demi-arrêt suivante j’ajoute l’angle de la demi-arrêt suivante et je mis à jour la demi-arrêt suivante par ça suivante jusqu’à revenir à la première face (indice de la demi-arrêt de la première demi arrête égale à l’indice de la suivante).

```
C:\Users\abdel\PycharmProjects\halfedge_mesh-master\venv\Scripts\python.exe C:/Users/abdel/PycharmProjects/halfedge_mesh-master/tp.py
[[1.5707963267948966, 1.5707963267948966, 0], [1.5707963267948966, 0, 1.5707963267948966], [1.5707963267948966, 0, 1.5707963267948966], [1.570

Process finished with exit code 0
```

Le test pour obtenir Les angles diédraux (3 angles pour chaque face) du fichier cube.off avant d’appliquer les approches :

Utilisation des approches :

- Angle maximum (norme infinis) :

```
def max_angle(mesh):
    valeurs=[]
    angle=angle_ddrl(mesh)
    for i in range(len(angle)):
        max_angle=max(angle[i])
        valeurs.append(max_angle)
    return valeurs
```

Après la récupération des angles diédraux de chaque face (3 angles), j'ai essayé de prendre un angle maximal pour chaque 3 angles d'une surface.

```
C:\Users\abdel\PycharmProjects\halfedge_mesh-master\venv\Scripts\python.exe C:/Users/abdel/PycharmPr
[1.5707963267948966, 1.5707963267948966, 1.5707963267948966, 1.5707963267948966, 1.5707963267948966,

Process finished with exit code 0
```

Le résultat pour le fichier cube.off l'algorithme a pris que les angles maximaux pour chaque face.

- Angle moyenne (norme 1) :

```
def moyenne_angle(mesh):
    valeurs = []
    angle = angle_ddrl(mesh)
    for i in range(len(angle)):
        moyenne_angle = mean(angle[i])
        valeurs.append(moyenne_angle)
    return valeurs
```

Pareil que l'approche de l'angle maximum, juste ici l'algorithme calcule la moyenne des 3 angles d'une face et la mettre dans une nouvel liste valeur.

```
C:\Users\abdel\PycharmProjects\halfedge_mesh-master\venv\Scripts\python.exe C:/Users/abdel/PycharmPr
[1.0471975511965976, 1.0471975511965976, 1.0471975511965976, 1.0471975511965976, 1.0471975511965976,

Process finished with exit code 0
```

Le résultat pour le fichier cube.off l'algorithme a pris la moyenne pour chaque 3 angles.

2-Visualisation de la propriété locale :

Création d'une fonction qui calcule le max et le min du tableau valeurs et qui converti chaque valeur en une couleur encodée en RGBA, par exemple pour une valeur plus petite la couleur blanche et la couleur rouge pour la plus grande valeur.

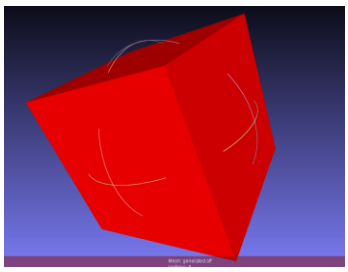
Fonction linéaire qui fixe R et Alpha et qui varie G, B linéairement.

```
def colors(mesh):
    couleur=[]
    #moyenne des angles
    #valeurs=moyenne_angle(mesh)
    #moyenne ameliorer en fonction des voisines
    valeurs=amelioration_loc(mesh)
    maxi= max(valeurs)
    minn= min(valeurs)
    print(valeurs)
    for i in range(len(valeurs)):
        if valeurs[i]==maxi:
            couleur.append([1.0, 0.0,0.0,1.0])
        elif valeurs[i]==minn:
            couleur.append([1.0, 1.0, 1.0, 1.0])
        else:
            #amelioration lineaire G et B varie
            couleur.append([1.0, 1 -(valeurs[i]/maxi), 1 -(valeurs[i]/maxi), 1.0])

    return couleur
```

Une fonction pour convertir la liste des couleurs en string pour l'utiliser après dans la création du fichier off, concaténation des champs (faces et couleurs).

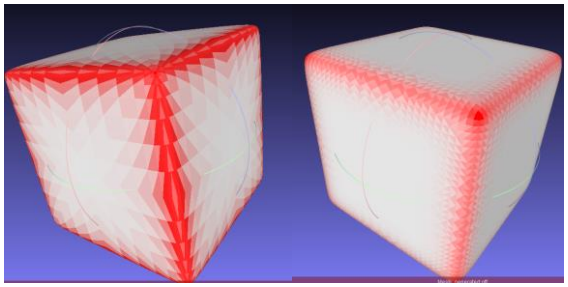
```
def couleur_conversion(mesh):
    col=[]
    fin=[]
    #couleur=colors(mesh)
    couleur=segmentation(mesh)
    for i in couleur:
        col=' '.join(str(elem) for elem in i)
        fin.append(col)
    return fin
```

Teste de coloriage pour le fichier cube.off :

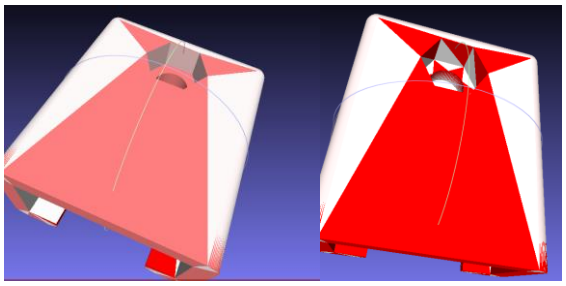


Les faces ont la même angle diéadrale cas d'un cube normal, c'est pour ça on a tous les faces avec la même couleur.

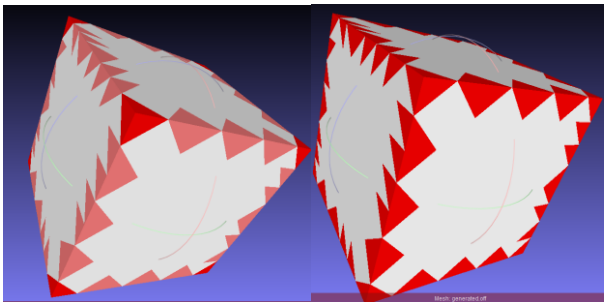
Pour le fichier Cube-smooth.off (par angle maximal et par :



Pour le fichier Adapter.off (par angle moyenne et angle maximal) :



Pour le fichier Cube_highres (par angle moyenne, par angle maximal) :



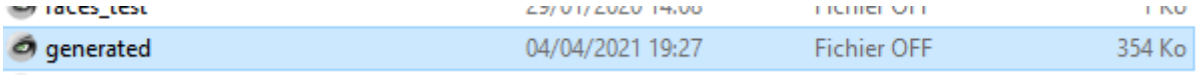
- Fabrication de fichier OFF :

La structure d'un fichier OFF se compose du mot OFF en haut du fichier ensuite le nombre des sommets et le nombre des facets ainsi que le nombre des arrêtes qui est le nombre des demi-arrêts sur 2, après on prend les coordonnées des sommets (x, y, z) et puis 3 car on a des

```
def create_off_file(mesh):
    couleurs=couleur_conversion(mesh)
    file_content=['OFF','\n'+str(len(mesh.vertices))+' '+str(len(mesh.facets))+' '+str(int(len(mesh.halfedges)/2))]
    for vertex in mesh.vertices:
        s='\n'+str(vertex.x)+' '+str(vertex.y)+' '+str(vertex.z)
        file_content.append(s)
    for face in range(len(mesh.facets)):
        s='\n3 '+str(mesh.facets[face].a)+' '+str(mesh.facets[face].b)+' '+str(mesh.facets[face].c)+' '+couleurs[face]
        file_content.append(s)
    file=open('tests/data/generated.off','w')
    file.writelines(file_content)
    file.close()
```

faces triangulaires ainsi que les index des sommets qui compose cette face et à la fin la couleur associée à cette face.

Un fichier généré avec le nom generated.off :

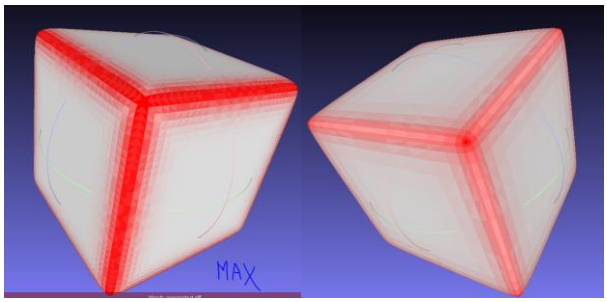


3-Amélioration de la propriété locale :

```
def amelioration_loc(mesh):
    valeur=moyenne_angle(mesh)
    nouveaux=[]
    for face in mesh.facets:
        valeur_facet = []
        halfedge=face.halfedge
        suivante_halfedge=halfedge.next
        opposit = halfedge.opposite
        index_voisine_facet = opposit.facet.index
        valeur_facet.append(valeur[index_voisine_facet])
        while halfedge.index!=suivante_halfedge.index:
            opposit=suivante_halfedge.opposite
            index_voisine_facet=opposit.facet.index
            valeur_facet.append(valeur[index_voisine_facet])
            suivante_halfedge = suivante_halfedge.next
        nouveaux.append(mean(valeur_facet))
    return nouveaux
```

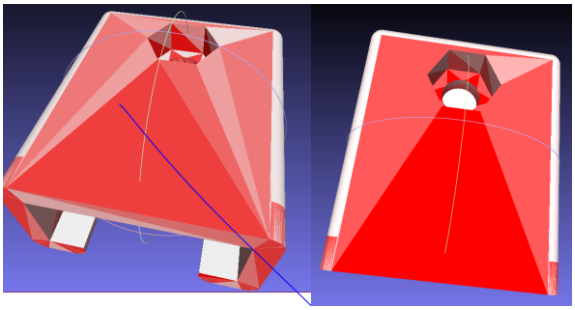
La fonction amélioration consiste à obtenir un nouveau tableau valeurs où chaque face a une valeur calculée comme moyenne des valeurs de ses faces voisines.

Pour le fichier Cube-smooth.off (par angle maximal, par angle moyenne):

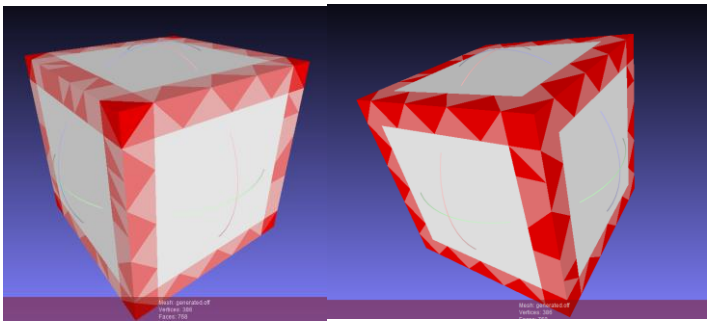


On remarque que sur les faces l'angle diéadrale et beaucoup plus proche du 0 (couleur dégradée) et dès qu'on avance vers les coins la valeur d'angle commence à s'augmenter (couleur foncée).

Pour le fichier Adapter.off (par angle moyenne, par angle maximal):



Pour le fichier Cube_highres (par angle moyenne et angle maximal):



Angle diédrale est augmenté sur les coins c'est pour ça la couleur des coins est foncée.

4-Segmentation en deux classes :

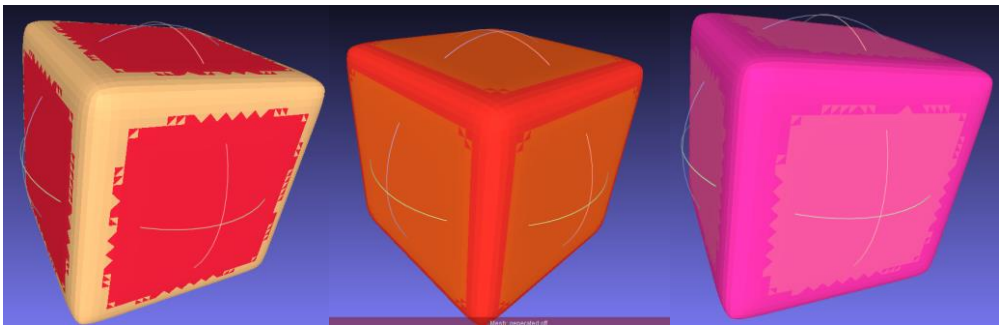
Cette partie réalisée pour l'obtention de deux classes de faces en se basant sur un seuil prédéfini (calculé sois par moyenne ou bien médiane ou bien fixé par l'utilisateur).

```
def seuil_par_pi():
    seuil =(math.pi)/36
    return seuil

def seuil_par_moyenne(mesh):
    moyenne = amelioration_loc(mesh)
    seuil = mean(moyenne)
    return seuil

def seuil_par_mediane(mesh):
    moyenne= amelioration_loc(mesh)
    seuil=np.median(moyenne)
    return seuil

def seuil_fixer_par_user():
    entrer=input("entrer un seuil")
    print(entrer)
```

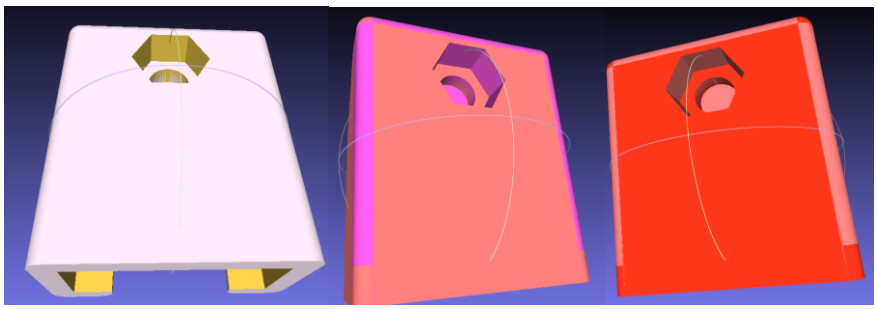
Les différents seuils utilisés :

```python
def segmentation(mesh):
    col=[]
    col1_classe=np.random.uniform(0,1,4)
    col2_classe=np.random.uniform(0,1,4)
    seuil=seuil_par_moyenne(mesh)
    seuil=seuil_par_mediane(mesh)
    #seuil=seuil_par_pi()
    valeurs=amelioration_loc(mesh)
    for val in valeurs:
        if val>seuil:
            col1_classe[0]=1
            col1_classe[3]=1
            col.append(col1_classe)
        else:
            col2_classe[0]= 1
            col2_classe[3]= 1
            col.append(col2_classe)
    return col
```

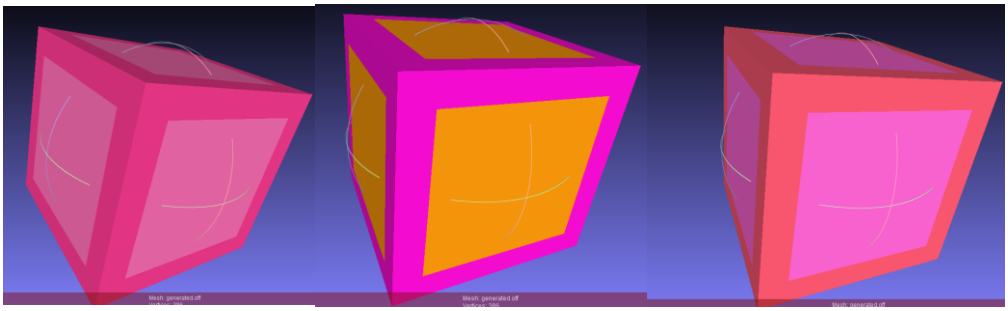
Pour le fichier Cube-smooth.off (par seuil fixé = 0.02, par moyenne et par médiane) :



Pour le fichier Adapter.off (par seuil fixé=0.02, par moyenne et par médiane) :

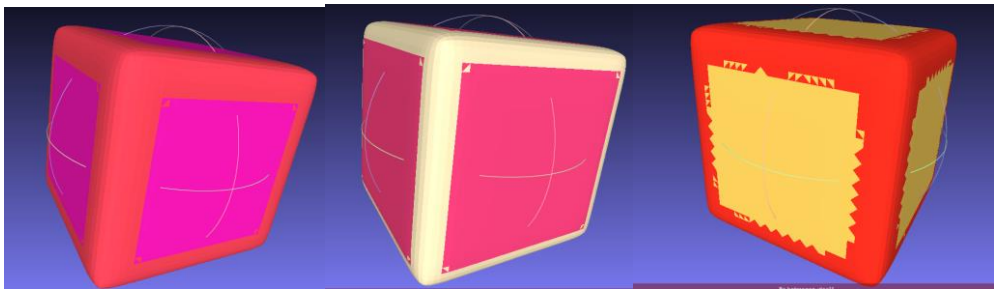


Pour le fichier Cube_highres par seuil fixé=0.02, par moyenne et par médiane) :

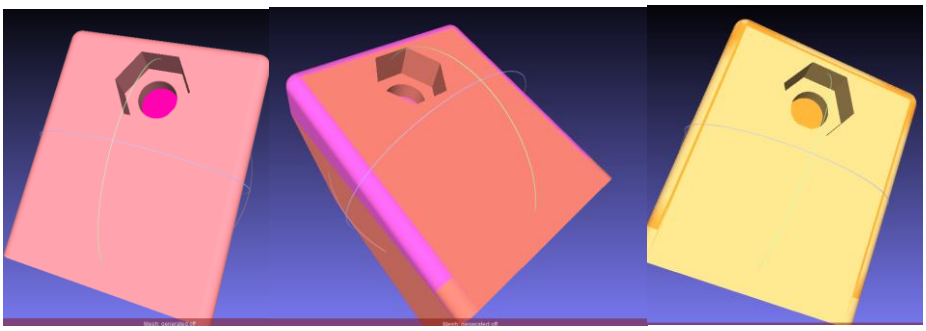


Par angle maximal :

Pour le fichier Cube-smooth.off (par seuil fixé = 0.02, par moyenne et par médiane) :



Pour le fichier Adapter.off (par seuil fixé=0.02, par moyenne et par médiane) :



Pour le fichier Cube_highres par seuil fixé=0.02, par moyenne et par médiane) :