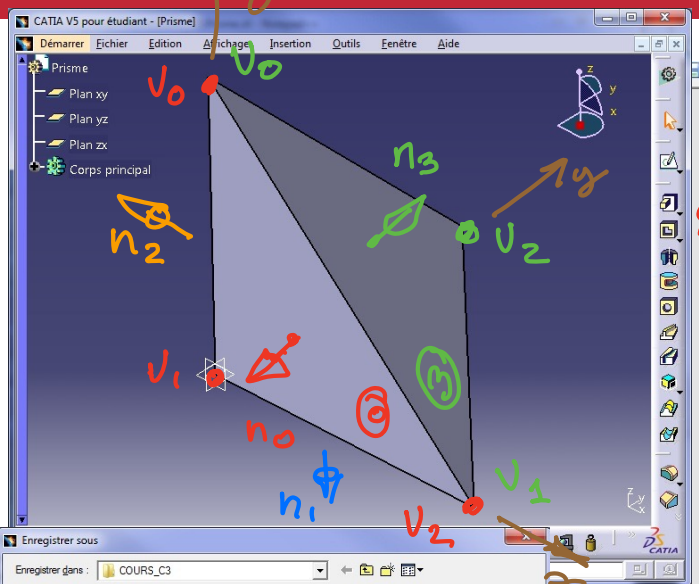
COURS C3 : TRAITEMENT DE FICHIER STL

Génération d'un fichier STL avec Catia



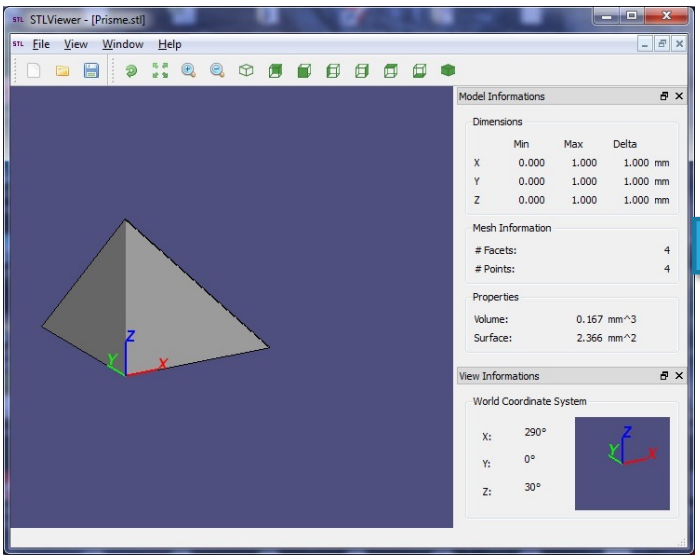
Prisme.stl

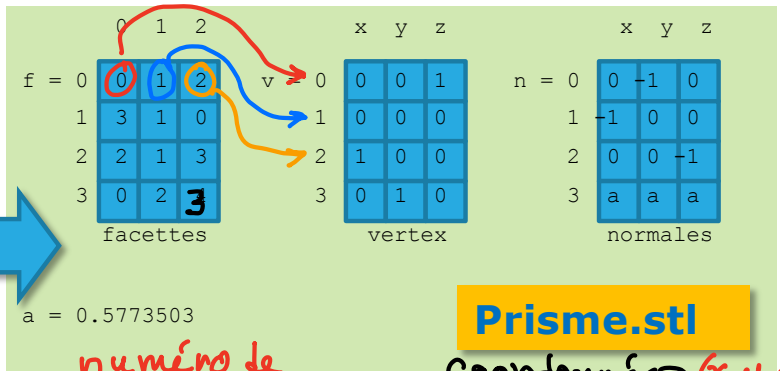
```
1  solid CATIA STL
2    facet normal  0.000000e+000 -1.000000e+000  0.000000e+000
3      outer loop
4        vertex  0.000000e+000  0.000000e+000  1.000000e+000
5        vertex  0.000000e+000  0.000000e+000  0.000000e+000
6        vertex  1.000000e+000  0.000000e+000  0.000000e+000
7      endloop
8    endfacet
9    facet normal -1.000000e+000  0.000000e+000  0.000000e+000
10     outer loop
11       vertex  0.000000e+000  1.000000e+000  0.000000e+000
12       vertex  0.000000e+000  0.000000e+000  0.000000e+000
13       vertex  0.000000e+000  0.000000e+000  1.000000e+000
14     endloop
15   endfacet
16   facet normal  0.000000e+000  0.000000e+000 -1.000000e+000
17     outer loop
18       vertex  1.000000e+000  0.000000e+000  0.000000e+000
19       vertex  0.000000e+000  0.000000e+000  0.000000e+000
20       vertex  0.000000e+000  1.000000e+000  0.000000e+000
21     endloop
22   endfacet
23   facet normal  5.773503e-001  5.773503e-001  5.773503e-001
24     outer loop
25       vertex  0.000000e+000  0.000000e+000  1.000000e+000
26       vertex  1.000000e+000  0.000000e+000  0.000000e+000
27       vertex  0.000000e+000  1.000000e+000  0.000000e+000
28     endloop
29   endfacet
30 endsolid CATIA STL
```

4 Facettes ⇒ 4 vertex unique

COURS C3 : TRAITEMENT DE FICHIER STL

Lecture d'un fichier STL

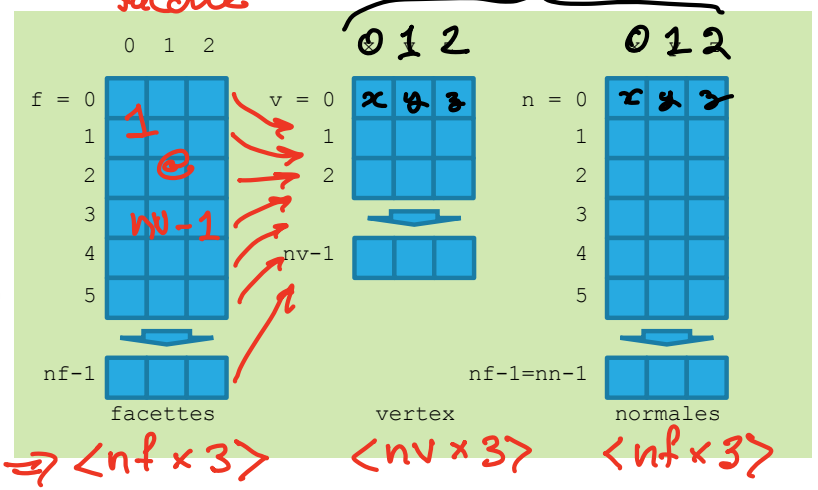




En général

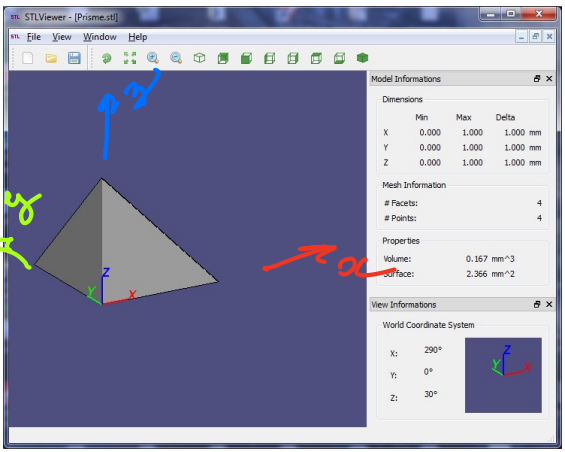
```
f , v , n = LireSTL(nom_fichier_in)
```

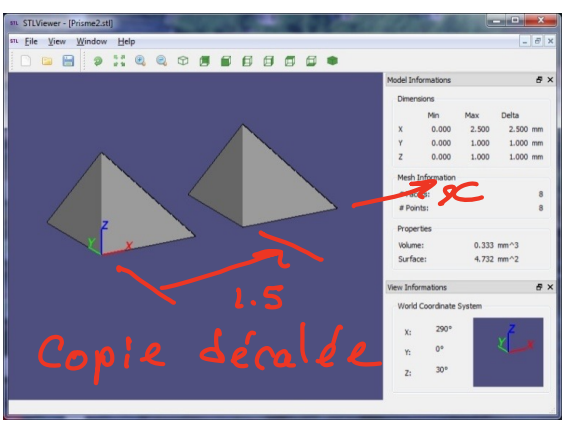
Fourni dans MEC1315_STL.py



COURS C3 : TRAITEMENT DE FICHIER STL

Ajout d'une copie à x=1.5






```
1   #Exemple C3_E1
2   import numpy as np
3   import matplotlib.pyplot as plt
4   from MEC1315_STL import *   #On importe tous les fonctions
5   nom_fichier_in="Prisme.stl"
6   f1,v1,n1=LireSTL(nom_fichier_in)
7   #=================================================
8   nv1=len(v1)            #nombre de points dans v1
9   f2=f1+nv1              #facettes f1+nv1
10  #=========Translation en x=========================
11  # v2=v1+[1.5,0,0]
12  #====Méthode 2 et 3 non fonctionnel==============
13  v2=np.array(v1)      #fonctionne
14  #v2=v1                #ne fonctionne pas
15  v2[:,0]=v2[:,0]+1.5
16  n2=n1
17  #=========Fusion===================================
18  f=np.vstack((f1,f2))
19  v=np.vstack((v1,v2))
20  n=np.vstack((n1,n2))
21  #============Écriture=============================
22  nom_fichier_out="Prisme2.stl"
23  EcrireSTLASCII(nom_fichier_out,f,v,n)
24
25  #======Traçage du graphique=======================
26  fig = plt.figure()
27  ax = fig.add_subplot(1, 1, 1, projection='3d')
28  ax.plot_trisurf(v[:,0],v[:,1],v[:,2], triangles=f, color='r')
```

← OK mais...

Fusion des 2 STL.

		0	1	2
f=	0	0	1	2
	1	3	1	0
f1	2	2	1	3
	3	0	2	3
	4	4	5	6
	5	7	5	4
f2	6	6	5	7
	7	4	6	7

		x	y	z
v=	0	0	0	1
	1	0	0	0
v1	2	1	0	0
	3	0	1	0
	4	1.5	0	1
	5	1.5	0	0
v2	6	2.5	0	0
	7	1.5	1	0

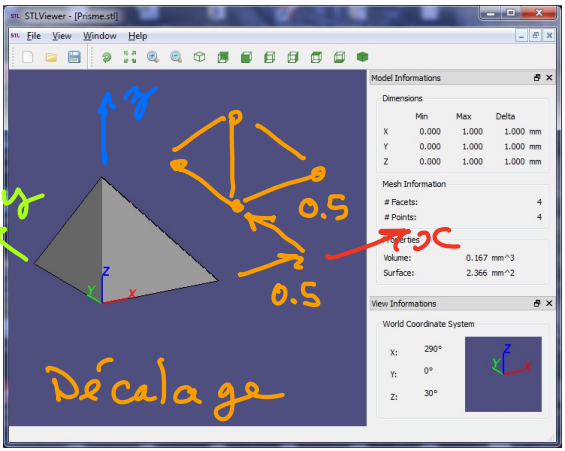
		x	y	z
n=	0	0	-1	0
	1	-1	0	0
n1	2	0	0	-1
	3	a	a	a
	4	0	-1	0
	5	-1	0	0
n2	6	0	0	-1
	7	a	a	a

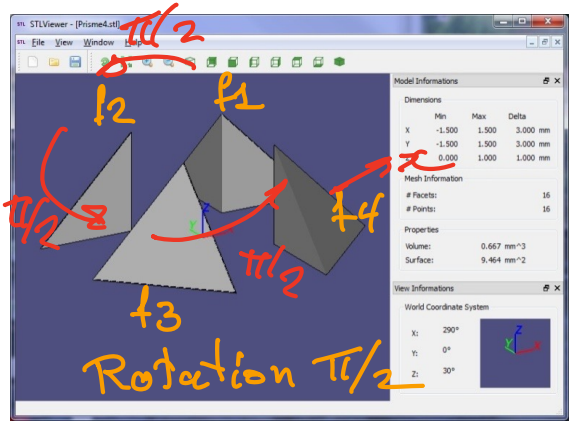
f2=f1+4 v2=v1+déplacement n2=n1 (pas de rotation)

⇒ F ⇒ ne change pas avec T, R, H
n ⇒ ne change pas avec T

COURS C3 : TRAITEMENT DE FICHIER STL

Translation à (0.5,0.5,0), puis 4 rotations en z de pi/2






```python
#Exemple C3_E1
import numpy as np
import matplotlib.pyplot as plt
from MEC1315_STL import *   #On importe tous les fonctions
nom_fichier_in="Prisme.stl"
f1,v1,n1=LireSTL(nom_fichier_in)
#====================================================
nv1=len(v1)
R=Rz(np.pi/2)
v1[:,0]=v1[:,0]+0.5         #Translation en x
v1[:,1]=v1[:,1]+0.5         #Translation en y
f2, v2, n2= f1+nv1, v1.dot(R), n1.dot(R) #Copie de 1 dans 2 avec rotation
f3, v3, n3=f2+nv1, v2.dot(R), n2.dot(R)  #Copie de 2 dans 3 avec rotation
f4, v4, n4=f3+nv1, v3.dot(R), n3.dot(R)  #Copie de 3 dans 4 avec rotation

#============Fusion==========================
f=np.vstack((f1,f2,f3,f4))
v=np.vstack((v1,v2,v3,v4))
n=np.vstack((n1,n2,n3,n4))
#==============Écriture=============================
nom_fichier_out="Prisme_rotation.stl"
EcrireSTLASCII(nom_fichier_out,f,v,n)

#======Traçage du graphique=========================
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1, projection='3d')
ax.plot_trisurf(v[:,0],v[:,1],v[:,2], triangles=f, color='r')
```

Fusion des 4 STL.

Nécessaire?

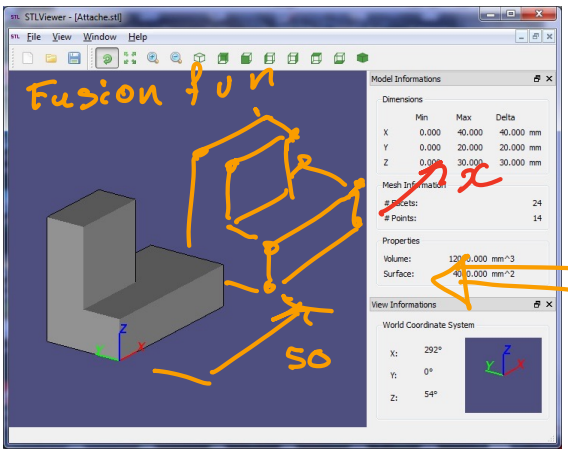
Fourni dans MEC1315_STL

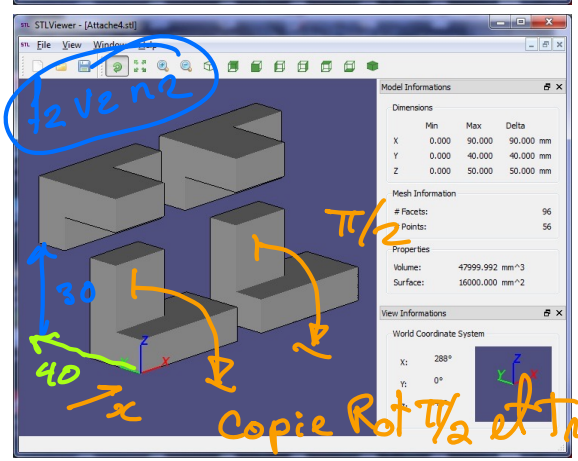
```python
def Rz(angle):
    Rz=np.array([[np.cos(angle),  np.sin(angle),  0],
                 [-np.sin(angle), np.cos(angle),  0],
                 [0            , 0            ,  1]])
    return Rz
```

et aussi Rx() et Ry()

COURS C3 : TRAITEMENT DE FICHIER STL

Copie en x, puis rotation et translation (0,40,30) du groupe






```python
#Exemple C3_E1
import numpy as np
import matplotlib.pyplot as plt
from MEC1315_STL import *   #On importe tous les fonctions
nom_fichier_in="Bloc.stl"
f1,v1,n1=LireSTL(nom_fichier_in)
#=================================================
nv1=len(v1)
f2=f1+nv1           #Copie de 1 dans 2
v2=np.array(v1)
v2[:,0]=v2[:,0]+50  #Translation en x de 50mm
n2=n1
#=============Fusion intermédiaire ================
f , v , n =np.vstack((f1,f2)) , np.vstack((v1,v2)) , np.vstack((n1,n2))
nv=len(v)
R=Rx(np.pi/2)       #Rotation autour de x de pi/2
f2 , v2 , n2 = f+nv , v.dot(R) , n.dot(R)   #Copie de 1 dans 2
v2[:,1]=v2[:,1]+40  #Translation de 40 unités en y
v2[:,2]=v2[:,2]+30  #Translation de 30 unités en z
#==============Fusion final=================
f=np.vstack((f,f2))
v=np.vstack((v,v2))
n=np.vstack((n,n2))
#==============Écriture=============================
nom_fichier_out="Bloc_modifie.stl"
EcrireSTLASCII(nom_fichier_out,f,v,n)
#======Traçage du graphique=========================
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1, projection='3d')
ax.plot_trisurf(v[:,0],v[:,1],v[:,2], triangles=f, color='r')
```

Fusion f v n + f2 v2 n2

```python
def Rx(angle):
    Rx=np.array([[1,     0      ,          0     ],
                 [0, np.cos(angle), np.sin(angle)],
                 [0, -np.sin(angle), np.cos(angle)]])
    return Rx

def Ry(angle):
    Ry=np.array([[np.cos(angle), 0, -np.sin(angle)],
                 [0           , 1 ,     0     ],
                 [np.sin(angle) ,0 ,    np.cos(angle)] ]  )
    return Ry
```

Fourni

COURS C3 : TRAITEMENT DE FICHIER STL

Lecture d'un fichier STL

Fichier de fonction MEC1315_STL fournit, doit être au même répertoire que le scripte de python ainsi que les fichiers STL

```python
def EcrireSTLASCII(output_file_name,face_fvn,vertex_fvn,normal_fvn):
    face_fvn=face_fvn.astype(int)
    f=open(output_file_name,"w")
    f.write("solid python")
    for j in range(len(face_fvn)):
        t1="facet normal"
        t2="outer loop"
        t3="endloop"
        t4="endfacet"
        f.write("\n  %s  %7.6e  %7.6e  %7.6e" % (t1, normal_fvn[j,0],normal_fvn[j,
        f.write("\n    %s" % t2)   #Écrire outerloop
        f.write("\n      vertex  %7.6e %7.6e %7.6e" % (vertex_fvn[face_fvn[j,0],0],
        f.write("\n      vertex  %7.6e %7.6e %7.6e" % (vertex_fvn[face_fvn[j,1],0],
        f.write("\n      vertex  %7.6e %7.6e %7.6e" % (vertex_fvn[face_fvn[j,2],0],
        f.write("\n    %s" % t3) #ecrire endloop
        f.write("\n  %s" % t4) #ecrire endfacet
    f.write("\n%s" % "Endsolid Python")
    f.close()
```

Fonction Écriture, formatage d'un fichier STL

```python
def Rx(angle):
    Rx=np.array([[1,     0      ,          0     ],
                 [0, np.cos(angle), np.sin(angle)],
                 [0, -np.sin(angle), np.cos(angle)]])
    return Rx

def Ry(angle):
    Ry=np.array([[np.cos(angle), 0, -np.sin(angle)],
                 [0           , 1 ,     0    ],
                 [np.sin(angle) ,0 ,    np.cos(angle)] ]  )
    return Ry
```

Fonction Rotation

Les normales peuvent être calculé avec f et v

```python
def CalculNormal(face_fvn,vertex_fvn):
    face_fvn=face_fvn.astype(int)
    normal_fvn=np.empty(np.shape(face_fvn))
    for i in range(len(face_fvn)):
        # print(f[i])
        vecteurA=vertex_fvn[face_fvn[i],0]
        vecteurB=vertex_fvn[face_fvn[i],1]
        vecteurC=vertex_fvn[face_fvn[i],2]
        vecteur_normal=np.cross((vecteurB-vecteurA), (vecteurC-vecteurA))
        normal_fvn[i]=vecteur_normal/np.linalg.norm(vecteur_normal)
    return normal_fvn
```

Fonction qui calcul la normale, utilisé pour les changements d'échelle par affinité vectorielle

COURS C3 : TRAITEMENT DE FICHIER STL

Écriture d'un fichier STL

```python
def LireSTL(nom_fichier):
    vertex_fvn, face_fvn =ReadSTL(nom_fichier)
    normal_fvn=GetNormals(vertex_fvn, face_fvn)
    return face_fvn, vertex_fvn, normal_fvn
```

Fonction LireSTL adapté pour le cours MEC1315

```python
def ReadSTL(filename):
    """ Returns numpy arrays for vertices and facet indexing """
    def GetListFromASCII(filename):
        """ Returns vertex listing from ASCII STL file """
        outputList=[]

        with open(filename,'r') as f:
            lines=[line.split() for line in f.readlines()]
        for line in lines:
            if line[0] == 'vertex':
                    outputList.append(tuple([float(x) for x in line[1:]]))
        return outputList

    def GetListFromBinary(filename):
        """ Returns vertex listing from binary STL file """
        outputList=[]
        with open(filename,'rb') as f:
            f.seek(80) # skip header
            nFacets=struct.unpack('I',f.read(4))[0] # number of facets in piece

            for i in range(nFacets):
                  f.seek(12,1) # skip normal
                  outputList.append(struct.unpack('fff',f.read(12))) # append each
                  outputList.append(struct.unpack('fff',f.read(12)))
                  outputList.append(struct.unpack('fff',f.read(12)))
                  f.seek(2,1) # skip attribute
        return outputList

    if IsBinarySTL(filename):
        vertexList = GetListFromBinary(filename)
    else:
        vertexList = GetListFromASCII(filename)

    coords, tempindxs = Unique(vertexList)

    indxs = list()
    templist = list()
    for i in range(len(tempindxs)):
        if (i > 0 ) and not (i % 3):
            indxs.append(templist)
            templist = list()
        templist.append(tempindxs[i])
    indxs.append(templist)

    return np.array(coords), np.array(indxs)
```

Si fichier débute par
solid ⇒ ascii
sinon ⇒ binaire

Fonction ReadSTL compatible pour fichier binaire et ascii. Code libre et tiré en ligne