<html><head><style>.pkt_added {text-decoration:none !important;}</style></head><body><pre
style="word-wrap: break-word; white-space: pre-wrap;">
```
import matplotlib.pyplot as plt
import matplotlib.lines as lines
import numpy as np
import os



def clean_up_strings(str_list):
    #remove any newlines from the end of strings
    while (str_list[len(str_list)-1]=="\n"):
        str_list.pop()
    for i in range(len(str_list)):
        str_list[i] = str_list[i].strip()

def read_obj_file(filename):
    sfile=open(filename,'r')
    l = sfile.readlines()
    clean_up_strings(l)
    vertices=[]
    triangles=[]
    for ln in l:
        strlist = ln.split()
        if (len(strlist)&gt;0):
            if (strlist[0] == "v"):
                v = [float(strlist[1]),float(strlist[2]),float(strlist[3])]
                vertices.append(v)
            elif (strlist[0] == "f") or (strlist[0] == "t"):
                t = [int(strlist[1])-1,int(strlist[2])-1,int(strlist[3])-1]
                triangles.append(t)
    sfile.close()
    return np.array(vertices), np.array(triangles)


def draw_model_lines(faces,vertices,normals,view,ax):
    for i in range(len(faces)):
        xs=[]
        ys=[]
        f=faces[i]
        if (not is_backfacing(n[i],view)):
            for j in range(4):
                vidx=f[j%3]
                xs.append(vertices[vidx][0])
                ys.append(vertices[vidx][1])
            ax.add_line(lines.Line2D(xs, ys, linewidth=1, color='black'))



def normalize_v3(arr):
    ''' Normalize a numpy array of 3 component vectors shape=(n,3) '''
    lens = np.sqrt( arr[:,0]**2 + arr[:,1]**2 + arr[:,2]**2 )
    arr[:,0] /= lens
    arr[:,1] /= lens
    arr[:,2] /= lens
    return arr

# Fill in these functions

def is_backfacing(n1,n2):
```

```
    pass

def draw_silhouette(faces,vertices,normals,view,ax):
   pass

#####

abspath = os.path.abspath(__file__)
dname = os.path.dirname(abspath)
os.chdir(dname)

vertices,faces = read_obj_file("cow.obj")

#Create a zeroed array with the same type and shape as our vertices i.e., per vertex
normal
norm = np.zeros( vertices.shape, dtype=vertices.dtype )
#Create an indexed view into the vertex array using the array of three indices for
triangles
tris = vertices[faces]
#Calculate the normal for all the triangles, by taking the cross product of the vectors
v1-v0, and v2-v0 in each triangle
n = np.cross( tris[::,1 ] - tris[::,0]  , tris[::,2 ] - tris[::,0] )
# n is now an array of normals per triangle. The length of each normal is dependent the
vertices,
# we need to normalize these, so that our next step weights each normal equally.
normalize_v3(n)


fig, ax = plt.subplots()
#draw_model_lines(faces,vertices,n,np.array([0,0,-1]),ax)
draw_silhouette(faces,vertices,n,np.array([0,0,-1]),ax)
fig.set_size_inches(6,6)          # Make graph square
plt.scatter([-0.1],[-0.1],s=0.01)     # Move graph window a little left and down


plt.plot()
plt.show()



</pre></body></html>
```