Olga Ibáñez Solé
Msc en Bioinformatique et Modélisation
NetID: oibanez
Numéro de matricule: 000426526
Groupe 1

PROJET 2: Vasarely doit se retourner dans sa tombe

```
import turtle
from math import cos, sin, pi

EPS = 1.0e-5

def deformation(p, c, r):

    """reçoit le point p = (x_p,y_p,z_p), le centre c = (x_c,y_c,z_c),
         et le rayon r tout les trois avant déformation,
         renvoit un point p2 après déformation
    """
    def distance(p,q):
        """distance entre les 2 points p et q"""
        return ((q[0]-p[0])**2 + (q[1]-p[1])**2 + (q[2]-p[2])**2)**0.5
    d = distance(p,c)

    if d >= r:
        res = p
    else:
        if d > EPS:
            facteur = r/d
        else:
            facteur = 1
        x2 = c[0] + (p[0]-c[0]) * facteur
        y2 = c[1] + (p[1]-c[1]) * facteur
        z2 = c[2] + (p[2]-c[2]) * facteur
        res = (x2, y2, z2)
    return res

deform = lambda p: deformation(p, centre, r)

def hexagone(t, c, longueur, col1, col2, col3, deform):
    """ reçoit turtle, le centre c = (x_c,y_c,z_c), la longueur entre le
    centre et n'importe quel coin de l'hexagone, les trois couleurs
    col1, col2 et col3 et la fonction deform,
```

```
renvoit un hexagone formé avec trois losanges peints de ces
trois couleurs
"""

"""Les six coins de l'hexagone (appelées "un", "deux",... "six") et
le point c (centre) sont deformés en leur applicant la fonction "deform"
"""
un = deform((c[0]+longueur, c[1], 0))
deux = deform((c[0]+longueur*(1-cos(pi/3)), c[1]-longueur*sin(pi/3), 0))
trois = deform((c[0]-longueur*cos(pi/3), c[1]-longueur*sin(pi/3), 0))
quatre = deform((c[0]-longueur, c[1], 0))
cinq = deform((c[0]-longueur*cos(pi/3), c[1]+longueur*sin(pi/3), 0))
six = deform((c[0]+longueur*(1-cos(pi/3)), c[1]+longueur*sin(pi/3), 0))
c = deform(c)

# Premier losange: inferieur droit.
t.up()
t.color(col1)
t.begin_fill()
t.goto(c[:2])
t.down()
t.goto(un[:2])
t.goto(deux[:2])
t.goto(trois[:2])
t.goto(c[:2])
t.end_fill()
t.up()

# Deuxième losange: gauche.
t.color(col2)
t.begin_fill()
t.goto(trois[:2])
t.down()
t.goto(quatre[:2])
t.goto(cinq[:2])
t.goto(c[:2])
t.end_fill()
t.up()
```

```
# Troisième losange: superieur droit.
   t.color(col3)
   t.begin_fill()
   t.goto(cinq[:2])
   t.down()
   t.goto(six[:2])
   t.goto(un[:2])
   t.goto(c[:2])
   t.end_fill()
   t.up()

def pavage(t, inf_gauche, sup_droit, longueur, col1, col2, col3, deform):

   """ reçoit turtle, les coordonées des coins inferieur gauche et supérieur
   droit de la fenêtre, la longueur entre le centre et n'importe quel coin
   de chaque hexagone, les trois couleurs col1, col2 et col3, et la fonction
   deform,
   renvoit la surface de la fenêtre avec des hexagones affichés, et dont une
   région circulaire de la fenêtre a été deformée
   """

   c = (inf_gauche, inf_gauche, 0)
   c_par = c
   c_impar = (inf_gauche + 1.5*longueur, inf_gauche + (longueur*sin(pi/3)),0)

   for i in range(int((abs(inf_gauche) + abs(sup_droit))//
(longueur*sin(pi/3)))):

        if i % 2 == 0: # Lignes ayant un indice pair
            c = (c_par[0], c_par[1] + i*longueur*sin(pi/3),0)
            for j in range((abs(inf_gauche) + abs(sup_droit))//(3*longueur)):
                hexagone(alex, c, longueur, col1, col2, col3, deform)
                c = (c[0] + 3*longueur, c[1],0)

        else: # Lignes ayant un indice impair
            c = (c_impar[0], c_impar[1] + (i-1)*(longueur*sin(pi/3)),0)

            for j in range((abs(inf_gauche) + abs(sup_droit))//(3*longueur)-1):
                hexagone(alex, c, longueur, col1, col2, col3, deform)
                c = (c[0] + 3*longueur, c[1],0)
```

```
"""
Arguments des fonctions deformation(), hexagone() et pavage() qui doivent être
lus sur input:
"""

inf_gauche = int(input("Coin inferieur gauche de la fenêtre = "))
sup_droit = int(input("Coin supérieur droit de la fenêtre = "))
longueur = int(input("Longueur entre le centre et n'importe quel coin de
l'hexagone = "))
r = int(input("Rayon du cercle = "))
centre = (int(input("Centre du cercle, x = ")), int(input("Centre du cercle, y =
")), int(input("Centre du cercle, z = ")))
col1 = input("Couleur 1 = ")
col2 = input("Couleur 2 = ")
col3 = input("Couleur 3 = ")
alex = turtle.Turtle()

# Execution de la fonction principale
pavage(alex, inf_gauche, sup_droit, longueur, col1, col2, col3, deform)
```