

Pavage de Penrose

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# référence : http://preshing.com/20110831/penrose-tiling-explained
# version un peu aménagée du travail de EC et LP, ba2 chimie 2012-2013

import math
import cmath
import cairo

# définir le nombre d'or
goldenRatio = (1 + math.sqrt(5)) / 2

def subdivide(triangles):
    result = []
    for color, A, B, C in triangles:
        if color == 0:
            # Subdiviser des petits triangles
            P = A + (B - A) / goldenRatio
            result += [(0, C, P, B), (1, P, C, A)]
        else:
            # Subdiviser des grands triangles
            Q = B + (A - B) / goldenRatio
            R = B + (C - B) / goldenRatio
            result += [(1, R, C, A), (1, Q, R, B), (0, R, Q, A)]
    return result

# Fonction définissant la lecture dans IDLE
def prompt(s, f):
    image = None
    while not image:
        entree = raw_input(s + '\n')
        image = f(entree)
    return image

# Convertit la taille
def convert_to_size(s):
    """Si la chaîne s est un couple d'entiers, retourne cet entier"""
    split = s.split(" ")
    if len(split) == 2:
        a = convert_to_positive_int(split[0])
        b = convert_to_positive_int(split[1])
        if a != None and b != None:
            return (a, b)
    return None

# Convertit e
```

```

def convert_to_positive_int(s):
    """Si la chaine s est un entier, retourne cet entier"""
    if s.isdigit() and long(s) > 0:
        return long(s)
    # Convertir en un retour en arriere
def convert_to_long(s):
    if s.isdigit():
        return long(s)
    # Convertit en couleur
def convert_to_color(s):
    """Si la chaine s est un triplet RGB, retourne ce triplet"""
    split = s.split(" ")
    print split
    if len(split) == 3:
        a = convert_to_long(split[0])
        b = convert_to_long(split[1])
        c = convert_to_long(split[2])
        if a != None and b != None and c != None:
            return (a / 255.0, b / 255.0, c / 255.0)
    return None

#----- Configuration -----
NUM_SUBDIVISIONS = prompt("Entrez le nombre de subdivisions desiree",
convert_to_positive_int)
IMAGE_SIZE = prompt("Entrez la taille d'image desiree, chaque composante
separee d'un espace", convert_to_size)
#-----

    # Creer une roue de petits triangles autour de l origine
triangles = []
for i in xrange(10):
    B = cmath.rect(1, (2*i - 1) * math.pi / 10)
    C = cmath.rect(1, (2*i + 1) * math.pi / 10)
    if i % 2 == 0:
        B, C = C, B # second triangle en miroir face a l autre
    triangles.append((0, 0j, B, C))
    # Ameliore les subdivisions
for i in xrange(NUM_SUBDIVISIONS):
    triangles = subdivide(triangles)
    # Prepare la surface cairo

surface = cairo.ImageSurface(cairo.FORMAT_ARGB32, IMAGE_SIZE[0],
IMAGE_SIZE[1])

cr = cairo.Context(surface)
cr.translate(IMAGE_SIZE[0] / 2.0, IMAGE_SIZE[1] / 2.0)
wheelRadius = 0.6 * math.sqrt((IMAGE_SIZE[0] / 2.0) ** 2 + (IMAGE_SIZE[1] /
2.0) ** 2)

```

```
cr.scale(wheelRadius, wheelRadius)
# Dessine le petit triangle
LittleTriangle = prompt("Entrez les composantes RGB du petit triangle
(strictement comprises entre 0 et 255), chaque composante separee d'un
espace", convert_to_color)
for color, A, B, C in triangles:
    if color == 0:
        cr.move_to(A.real, A.imag)
        cr.line_to(B.real, B.imag)
        cr.line_to(C.real, C.imag)
        cr.close_path()
cr.set_source_rgb(LittleTriangle[0], LittleTriangle[1], LittleTriangle[2])
cr.fill()
# Dessine le grand triangle
GreatTriangle = prompt("Entrez les composantes RGB du grand triangle
(strictement comprises entre 0 et 255), chaque composante separee d'un
espace", convert_to_color)
for color, A, B, C in triangles:
    if color == 1:
        cr.move_to(A.real, A.imag)
        cr.line_to(B.real, B.imag)
        cr.line_to(C.real, C.imag)
        cr.close_path()
cr.set_source_rgb(GreatTriangle[0], GreatTriangle[1], GreatTriangle[2])
cr.fill()
# Determine l epaisseur de la ligne du premier triangle
color, A, B, C = triangles[0]
cr.set_line_width(abs(B - A) / 10.0)
cr.set_line_join(cairo.LINE_JOIN_ROUND)
# Dessine la ligne
lineColor = prompt("Entrez les composantes RGB des segments de separation
(strictement comprises entre 0 et 255), chaque composante separee d'un
espace", convert_to_color)
for color, A, B, C in triangles:
    cr.move_to(C.real, C.imag)
    cr.line_to(A.real, A.imag)
    cr.line_to(B.real, B.imag)
cr.set_source_rgb(0.2, 0.2, 0.2)
cr.stroke()

surface.write_to_png('penrose.png')
```

From:

<https://dvillers.umons.ac.be/wiki/> - Didier Villers, UMONS - wiki

Permanent link:

https://dvillers.umons.ac.be/wiki/teaching:progappchim:pavage_penrose_2013

Last update: **2013/11/28 14:08**



