#! /usr/bin/python3

import os

import re

import random

skew=True # Skew the pattern generation to get more of the first pattern

letter=True # Print on letter paper (or on A4 if false)

patterns = ['chapeau\_bell', 'chapeau\_conical', 'ombre\_chapeau', 'colerette\_sans']

colors = ['green', 'orange', 'red', 'purple']

pmargin=1200 # page margin, in the unit of the viewBox

cmargin=645 # margin within the card

cellsize=2100 # cell of one icon, in the unit of the viewBox

cardsize=5 # Each card has cardsize x cardsize icons

xcard=3 # There is xcard x ycard on a given page

ycard=4 #

## compute some globals

paths = {} # <path> data drawing the pattern

# Each image is meant to be printed on a viewBoxes of a specific size: 512x512, 640x512 or 320x512 (check their svg header)

# so we need to compute extra padding to center them in the cell: xpad and ypad

xpad = {}

ypad = {}

random.seed(None)

## Read the image files and prepare the data

#regex to parse the svg

repath = re.compile("<path[^>]\*>")

rebox = re.compile('viewBox="0 0 ([^ ]\*) ([^"]\*)"')

recolor = re.compile('"/>')

for (filename,color) in zip(patterns,colors):

svgfile = open("svg\_parts/{:s}.svg".format(filename), "r")

svgstr = svgfile.read()

path = repath.search(svgstr).group()

# print('{:s}: {:s}\n'.format(filename, paths[filename]))

if not 'ombre' in filename and not 'points' in filename:

paths[filename] = recolor.sub('" fill="{:s}"/>'.format(color), path)

else:

paths[filename] = path

box = rebox.search(svgstr)

xpad[filename] = (cellsize - int(box.group(1))) / 2

assert xpad[filename] > 0, "the cell is too small for {:s}.svg: cellsize is {:d} but image is x{:s} y{:s}".format(filename, cellsize, box.group(1), box.group(2))

ypad[filename] = (cellsize - int(box.group(2))) / 2

assert ypad[filename] > 0, "the cell is too small for {:s}.svg: cellsize is {:d} but image is x{:s} y{:s}".format(filename, cellsize, box.group(1), box.group(2))

# Compute the data content

data = [[-1 for j in range(cardsize \* ycard)] for i in range(cardsize \* xcard)]

for xc in range(xcard):

for yc in range(ycard):

curamounts = [0 for i in range(len(patterns))]

for x in range(cardsize):

for y in range(cardsize):

if skew:

p = int(random.uniform(0,len(patterns)+2))

if p > (len(patterns)-1):

p=0

else:

p = int(random.uniform(0,len(patterns)))

curamounts[p] += 1

data[x+xc\*cardsize][y+yc\*cardsize] = p

print(curamounts, sum(curamounts))

# Helping function

def cell\_to\_viewport(pattern, x, y):

vx=pmargin + x\*cellsize + (0.5+int(x/(cardsize)))\*cmargin + xpad[pattern]

vy=pmargin + y\*cellsize + (0.5+int(y/(cardsize)))\*cmargin + ypad[pattern]

return 'x="{:f}" y="{:f}"'.format(vx, vy)

## Generate the file

f = open("board.svg", "w")

f.write('<svg xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink"\n')

if letter:

f.write(' viewBox="0 0 14000 19800" height="11in" width="8.5in">\n') # 14000x19800 is 21000x29700 at 2/3 ratio

else:

# f.write(' viewBox="0 0 14000 19800" height="297mm" width="210mm">\n') # 14000x19800 is 21000x29700 at 2/3 ratio

f.write(' viewBox="0 0 14000 19800" height="594mm" width="420mm">\n') # 14000x19800 is 21000x29700 at 2/3 ratio

f.write('<defs>\n')

for pat in patterns:

f.write(' <g id="{:s}">{:s}</g>\n'.format(pat, paths[pat]))

f.write('</defs>\n')

# Cells content

for x in range(cardsize \* xcard):

for y in range(cardsize \* ycard):

pat = patterns[ data[x][y] ]

f.write('<use xlink:href="#{:s}" {:s} />\n'.format(pat, cell\_to\_viewport(pat, x,y)))

# Grid to help cutting the cards

for x in range(xcard):

for y in range(ycard):

f.write('<rect x="{:f}" y="{:f}" width="{:f}" height="{:f}" {:s} />'

.format(pmargin+x\*cellsize\*(cardsize+1), pmargin+y\*cellsize\*(cardsize+1), cellsize\*(cardsize+1), cellsize\*(cardsize+1),

'style="stroke:#c4c4c4;stroke-width:3;stroke-opacity:1;fill:none"'))

f.write('</svg>\n')

f.close()

os.system("inkscape --export-pdf=board.pdf board.svg")

os.unlink("board.svg")