import matplotlib.pyplot as plt

import matplotlib.colors

import math

import numpy as np

class SqueezedNorm(matplotlib.colors.Normalize):

def \_\_init\_\_(self, vmin=None, vmax=None, mid=0, s1=2, s2=2, clip=False):

self.vmin = vmin # minimum value

self.mid = mid # middle value

self.vmax = vmax # maximum value

self.s1=s1; self.s2=s2

f = lambda x, zero,vmax,s: np.abs((x-zero)/(vmax-zero))\*\*(1./s)\*0.5

self.g = lambda x, zero,vmin,vmax, s1,s2: f(x,zero,vmax,s1)\*(x>=zero) - \

f(x,zero,vmin,s2)\*(x<zero)+0.5

matplotlib.colors.Normalize.\_\_init\_\_(self, vmin, vmax, clip)

def \_\_call\_\_(self, value, clip=None):

r = self.g(value, self.mid,self.vmin,self.vmax, self.s1,self.s2)

return np.ma.masked\_array(r)

def retrieve(filename):

f = open(filename, "r")

new = f.readlines()

points = []

for p in new:

x = []

z = p.replace("[", "").replace("]", "").replace(",", "").split()

x += [float(z[0])]

x += [float(z[1])]

x += [float(z[2])]

points += [x]

f.close()

return points

def translatewicket(value):

valueScaled = value / 360

return 0.5 + (valueScaled \* 24)

def translateele(value, min):

valueScaled = (value\*-1)-min

return -1\*(valueScaled\*39.71392+22.75)

def ang\_calc(point, center):

if point[0] > center[0] and point[1] > center[1]:

angle = math.degrees(math.atan(math.fabs((point[0] - center[0]) / (point[1] - center[1]))))

elif point[0] > center[0] and point[1] < center[1]:

angle = math.degrees(math.atan(math.fabs((point[1] - center[1]) / (point[0] - center[0])))) + 90

elif point[0] < center[0] and point[1] < center[1]:

angle = math.degrees(math.atan(math.fabs((point[0] - center[0]) / (point[1] - center[1])))) + 180

elif point[0] < center[0] and point[1] > center[1]:

angle = math.degrees(math.atan(math.fabs((point[1] - center[1]) / (point[0] - center[0])))) + 270

elif point[0] == center[0] and point[1] > center[1]:

angle = 0

elif point[0] == center[0] and point[1] < center[1]:

angle = 180

elif point[1] == center[1] and point[0] > center[0]:

angle = 90

elif point[1] == center[1] and point[0] < center[0]:

angle = 270

return angle

def map(filename, center):

minim = .12683

points = retrieve(filename)

heatmap = plt.figure()

plt.title("Radial Gap Heatmap (Unit 3 As Found)")

track = 0

x = []

y = []

colors = []

for p in points:

print(track)

track += 1

val = 1000 \* (math.sqrt(((center[0] - p[0]) \*\* 2) + ((center[1] - p[1]) \*\* 2)) - math.sqrt(

(center[3] \*\* 2) - ((center[2] - p[2]) \*\* 2)))

x\_axis = translatewicket(ang\_calc(p, center))

#test = translatehex(val, min, max)

#color = col\_list["hex"][test]

x += [x\_axis]

y += [translateele(p[2],minim)]

#colors += [color])

colors += [val/25.4]

# pcm = plt.pcolormesh(x, y, colors, vmin=-1., vmax=1., cmap='RdBu\_b')

print(max(colors))

print(min(colors))

norm = SqueezedNorm(vmin=0.11, vmax=2, mid=0.67, s1=0.89, s2=1.06)

plt.scatter(x, y, s=1, c=colors, cmap='gist\_rainbow', norm=norm)

plt.grid(b=True)

cbar = plt.colorbar()

cbar.ax.set\_ylabel('Radial Gap (inches)', rotation=270)

cbar.ax.yaxis.set\_label\_coords(3.8, 0.5)

plt.xlabel("Wicket Gate")

plt.ylabel("Distance From Horizontal Weld (inches)")

plt.show()

return

# these two parameters refer to the calculated center and the points to be mapped

center = [-1.0186031433217924, 9.356643518673076, -0.48031975059244286, 3.3223712382918738]

filename = "hmap\_points.txt"

map(filename, center)