# -\*- coding: utf-8 -\*-

from PyQt5.QtWidgets import (QWidget, QApplication, QGroupBox, QPushButton,

QLabel, QHBoxLayout, QVBoxLayout, QGridLayout, QFormLayout, QLineEdit, QTextEdit,QComboBox,QMessageBox,

QDesktopWidget, QFileDialog,QTextEdit, QMessageBox)

from PyQt5.QtGui import QPixmap, QIcon, QFont, QGuiApplication, QTextBlockFormat, QTextCursor

from matplotlib.backends.backend\_qt5agg import FigureCanvasQTAgg as FigureCanvas

from matplotlib.backends.backend\_qt5agg import NavigationToolbar2QT as NavigationToolbar

import matplotlib.pyplot as plt

import matplotlib.patches as mpatches

import matplotlib.path as mpath

from matplotlib.ticker import AutoMinorLocator, OldAutoLocator, ScalarFormatter,MaxNLocator

from matplotlib import ticker,axes,projections

from matplotlib.lines import lineStyles

from matplotlib.figure import Figure

from math import sqrt,pi,log10

import numpy as np

import sys

import random

import time

import os

import webbrowser

#使任务栏显示程序图标

import ctypes

ctypes.windll.shell32.SetCurrentProcessExplicitAppUserModelID(*"myappid"*)

class **MyCostumToolbar**(NavigationToolbar):

toolitems = [t for t in NavigationToolbar.toolitems if

t[0] in (*'Home'*, *'Back'*,*'Forward'*,*'Pan'*, *'Zoom'*,*'Save'*)]

class **My\_Axes**(axes.Axes):

name=*'My\_Axes'*

def **drag\_pan**(*self*,button,key,x,y):

axes.Axes.drag\_pan(*self*, button, *'xy'*, x, y)

class **Window**(QWidget):

def **\_\_init\_\_**(*self*):

super(Window,*self*).\_\_init\_\_()

*self*.creategridbox()

*self*.createpic()

*self*.createpic2()

*self*.pixmap=QPixmap(*'1.png'*)

*self*.lb=QLabel(*self*)

*self*.lb.setPixmap(*self*.pixmap)

*self*.hbox=QHBoxLayout()

*self*.vbox1=QVBoxLayout()

*self*.vbox2=QVBoxLayout()

*self*.vbox1.addWidget(*self*.lb)

*self*.vbox1.addWidget(*self*.picbox)

*self*.vbox2.addWidget(*self*.gridbox)

*self*.vbox2.addWidget(*self*.picbox2)

*self*.hbox.addLayout(*self*.vbox2)

*self*.hbox.addStretch(1)

*self*.hbox.addLayout(*self*.vbox1)

*self*.setLayout(*self*.hbox)

*self*.setWindowTitle(*'Pre-PPDEM V1.0'*)

*self*.setWindowIcon(QIcon(*'dragon.ico'*))

*self*.setGeometry(100, 50, 750,900)

*self*.center() #获取显示屏幕大小动态设置

*self*.x1=[]

*self*.y1=[]

*self*.dd=[] #随机分布后顺序半径

*self*.an=[]

*self*.cac=[]

*self*.coord=[]

*self*.infodic={}

*self*.num=0

*self*.ifplot=0

def **creategridbox**(*self*):

*self*.gridbox=QGroupBox()

grid=QGridLayout()

*'''*

*self.fig = Figure(figsize=(5, 10), dpi=100)*

*self.canvas=FigureCanvas(self.fig)*

*self.toolbar=MyCostumToolbar(self.canvas,self)*

*btn=QPushButton('Grading Curve')*

*btn.clicked.connect(self.plot)*

*'''*

*self*.btr=QPushButton(*'Read File'*)

*self*.btr.clicked.connect(*self*.readfile)

*self*.bth=QPushButton(*'Help'*)

*self*.bth.clicked.connect(*self*.helpfile)

*self*.TB1=QTextEdit(*'Grading'*)

*self*.TB1.setFixedSize(150,250)

#设置TextEdit行高

text\_format=QTextBlockFormat()

text\_format.setBottomMargin(0)

text\_format.setLineHeight(15,QTextBlockFormat.FixedHeight)

text\_cursor=*self*.TB1.textCursor()

text\_cursor.setBlockFormat(text\_format)

*self*.TB1.setTextCursor(text\_cursor)

Lbn=QLabel(*'Particle Number'*) #不少于100个

*self*.LEn=QLineEdit(*'200'*)

Lbr=QLabel(*'Specimen Slenderness'*)

*self*.LEr=QLineEdit(*'2'*)

Lbe=QLabel(*'Estimated Void Ratio'*)

*self*.LEe=QLineEdit(*'0.5'*)

*self*.LEn.setFixedWidth(100)

*self*.LEr.setFixedWidth(100)

*self*.LEe.setFixedWidth(100)

#set the layout

grid.addWidget(*self*.btr,0,0,1,1)

grid.addWidget(*self*.bth,0,6,1,1)

grid.addWidget(*self*.TB1,1,0,15,4)

grid.addWidget(Lbn,1,5,1,1)

grid.addWidget(*self*.LEn,1,6,1,1)

grid.addWidget(Lbr,2,5,1,1)

grid.addWidget(*self*.LEr,2,6,1,1)

grid.addWidget(Lbe,3,5,1,1)

grid.addWidget(*self*.LEe,3,6,1,1)

grid.setHorizontalSpacing(15)

*self*.gridbox.setLayout(grid)

*self*.gridbox.setWindowTitle(*'test'*)

def **createpic**(*self*):

*self*.picbox=QGroupBox()

hbox1=QHBoxLayout()

hbox2=QHBoxLayout()

vbox=QVBoxLayout()

*self*.fig2 = Figure(figsize=(5,30), dpi=100)

*self*.canvas2=FigureCanvas(*self*.fig2)

*self*.cb=QComboBox()

*self*.cb.addItems([*'circle'*,*'ellipse'*,*'triangle'*,*'rectangle'*,*'pentagon'*])

*self*.cb.currentIndexChanged.connect(*self*.comvisi)

toolbar=MyCostumToolbar(*self*.canvas2,*self*)

btn=QPushButton(*'Simulation'*)

btn.clicked.connect(*self*.plot2)

*self*.Lbc=QLabel(*'invisible'*)

*self*.Lbc.hide()

*self*.LEc=QLineEdit(*'2'*)

*self*.LEc.hide()

*self*.LEc.setFixedWidth(100)

*self*.Lbc2=QLabel(*'Roundness(0-1)'*)

*self*.Lbc2.hide()

*self*.LEc2=QLineEdit(*'0'*)

*self*.LEc2.hide()

*self*.LEc2.setFixedWidth(100)

btnp=QPushButton(*'Output'*)

btnp.clicked.connect(*self*.output)

hbox1.addWidget(toolbar)

hbox1.addWidget(*self*.cb)

hbox1.addWidget(btn)

hbox2.addWidget(*self*.Lbc)

hbox2.addWidget(*self*.LEc)

hbox2.addWidget(*self*.Lbc2)

hbox2.addWidget(*self*.LEc2)

hbox2.addStretch(1)

hbox2.addWidget(btnp)

vbox.addLayout(hbox1)

vbox.addLayout(hbox2)

vbox.addStretch(1)

vbox.addWidget(*self*.canvas2)

*self*.picbox.setLayout(vbox)

def **createpic2**(*self*):

*self*.picbox2=QGroupBox()

hbox1=QHBoxLayout()

vbox=QVBoxLayout()

*self*.fig = Figure(figsize=(5,5), dpi=100)

*self*.canvas=FigureCanvas(*self*.fig)

toolbar=MyCostumToolbar(*self*.canvas,*self*)

btn=QPushButton(*'Grading Curve'*)

btn.clicked.connect(*self*.plot)

hbox1.addWidget(toolbar)

hbox1.addWidget(btn)

vbox.addLayout(hbox1)

vbox.addStretch(1)

vbox.addWidget(*self*.canvas)

*self*.picbox2.setLayout(vbox)

def **helpfile**(*self*):

fo=open(webbrowser.open(*"readme.docx"*))

def **readfile**(*self*):

filename=QFileDialog.getOpenFileName(*self*, *'Open File Dialog'*, *'C:'*,*"Txt files(\*.txt)"*)

ch=[]

i=0

f=open(filename[0],*'r'*)

#重置输出框

*self*.TB1.setText(*'Grading\n d\tper'*)

#设置TextEdit行高

text\_format=QTextBlockFormat()

text\_format.setBottomMargin(0)

text\_format.setLineHeight(15,QTextBlockFormat.FixedHeight)

text\_cursor=*self*.TB1.textCursor()

text\_cursor.setBlockFormat(text\_format)

*self*.TB1.setTextCursor(text\_cursor)

try:

while True:

lines=f.readline()

if not lines:

break

d\_tmp,p\_tmp=[float(i) for i in lines.split()]

*self*.TB1.append(str(d\_tmp)+*'\t'*+str(p\_tmp)+*'\n'*)

*self*.infodic[d\_tmp]=p\_tmp

except:

QMessageBox.warning(*self*, *'Warning'*, *'Invalid Input Format'*,QMessageBox.Yes)

return

def **center**(*self*):

index=QDesktopWidget().primaryScreen()

screen = QDesktopWidget().availableGeometry(index)

size = *self*.frameGeometry()

*self*.move(screen.width()/2 - size.width()/1.5,

(screen.height() - size.height()) / 2)

def **output**(*self*):

if(*self*.ifplot==0):

*self*.plot2()

dir=None

try:

dir = QFileDialog.getExistingDirectory(*self*,

*"Select File Directory"*,

*"C:/"*) #起始路径

except:

return

if(dir==None):

return

note=(*"\t"* +*"4"*+*"\n"*+

*"-0.5825781033135710E+00 0.5574716361287161E+00 0.6345870624177348E+00 0.5574716361287161E+00"*+*"\n"*+

*"0.6345870624177348E+00 0.6162241217404306E+00 -0.5825781033135710E+00 0.6162241217404306E+00"*+*"\n"*+

*"-0.4172132111845057E-03 0.1198385011842817E+01 -0.4172132111845057E-03 -0.2468925397367039E-01"*+*"\n"*+

*"0.5242617231534830E-01 -0.2468925397367039E-01 0.5242617231534830E-01 0.1198385011842817E+01"*+*"\n"*+

*"0.3814697265625000E-05 0.2147483648000000E+10 0.1000000000000000E+01 0.5000000000000000E+00 0.1000000000000000E+01 0.1864399082996630E-03 0.0000000000000000E+00 0.0000000000000000E+00 0.0000000000000000E+00 0.8000000000000001E-04 0.0000000000000000E+00 0.1000000000000000E+02 0.0000000000000000E+00 0.2000000000000000E+03 0.0000000000000000E+00 0.5967993208922544E-02 0.0000000000000000E+00 0.2000000000000000E+00 0.2000000000000000E+00 0.2000000000000000E+00"*+*"\n"*+

*"dt E globDamping alpha beta maxGap tanTheta pX pY pInt gX gY xGravity yGravity CAC"*

)

if *self*.cb.currentText()==*'ellipse'*:

ra=int(*self*.LEc.text())

localtime=time.strftime(*"%Y-%m-%d %H.%M.%S"*,time.localtime())

fo=open(dir+*"/ellipse "*+localtime+*".txt"*,*"w"*)

fo.writelines(str(*self*.num)+*" "*+*"1"*+*"\n"*)

for i in range(0,*self*.x1.\_\_len\_\_()):

m=*self*.calelli(*self*.x1[i], *self*.y1[i], *self*.dd[i], *self*.dd[i]/ra, *self*.an[i])

str1=(*"4"*+*" "*+str(m[0][0,0])+*" "*

+str(m[0][0,1])+*" "*+str(m[1][0,0])+*" "*

+str(m[1][0,1])+*" "*+str(m[2][0,0])+*" "*

+str(m[2][0,1])+*" "*+str(m[3][0,0])+*" "*

+str(m[3][0,1])+*" "*+str(*self*.cac[0])

+*" "*+str(*self*.cac[1])+*" "*+str(*self*.cac[2])

+*" "*+str(*self*.cac[3])

)

fo.writelines(str1+*'\n'*)

fo.writelines(note+*'\n'*)

fo.close()

elif *self*.cb.currentText()==*'circle'*:

localtime=time.strftime(*"%Y-%m-%d %H.%M.%S"*,time.localtime())

fo=open(dir+*"/circle "*+localtime+*".txt"*,*"w"*)

fo.writelines(str(*self*.num)+*" "*+*"1"*+*"\n"*)

for i in range(0,*self*.x1.\_\_len\_\_()):

str1=*"2"*+*" "*+(str(*self*.x1[i]/1000)+*" "*+str(*self*.y1[i]/1000)+*" "*+str(*self*.dd[i]/2000)+*" "*+str(*self*.dd[i]/2000))

fo.writelines(str1+*'\n'*)

fo.writelines(note+*'\n'*)

fo.close()

elif *self*.cb.currentText()==*'triangle'*:

localtime=time.strftime(*"%Y-%m-%d %H.%M.%S"*,time.localtime())

fo=open(dir+*"/triangle "*+localtime+*".txt"*,*"w"*)

roundness=float(*self*.LEc2.text())

fo.writelines(str(*self*.num)+*" "*+*"1"*+*"\n"*)

for i in range(0,*self*.x1.\_\_len\_\_()):

str1=(*"3"*+*" "*+str(*self*.coord[i][0,0])+*" "*+str(*self*.coord[i][0,1])+*" "*+str(*self*.coord[i][1,0])+*" "*+

str(*self*.coord[i][1,1])+*" "*+str(*self*.coord[i][2,0])+*" "*

+str(*self*.coord[i][2,1])

)

if(roundness==0):

str1+=*" "*+*"0.1"*+*" "*+*"0.1"*+*" "*+*"0.1"*

else:

str1+=*" "*+str(roundness\**self*.cac[i][0])+*" "*+str(roundness\**self*.cac[i][1])+*" "*+str(roundness\**self*.cac[i][2])

fo.writelines(str1+*'\n'*)

fo.writelines(note+*'\n'*)

fo.close()

elif *self*.cb.currentText()==*'rectangle'*:

localtime=time.strftime(*"%Y-%m-%d %H.%M.%S"*,time.localtime())

fo=open(dir+*"/rectangle "*+localtime+*".txt"*,*"w"*)

roundness=float(*self*.LEc2.text())

fo.writelines(str(*self*.num)+*" "*+*"1"*+*"\n"*)

for i in range(0,*self*.x1.\_\_len\_\_()):

str1=(*"4"*+*" "*+str(*self*.coord[i][0,0])+*" "*+str(*self*.coord[i][0,1])+*" "*+str(*self*.coord[i][1,0])+*" "*+

str(*self*.coord[i][1,1])+*" "*+str(*self*.coord[i][2,0])+*" "*

+str(*self*.coord[i][2,1])+*" "*+str(*self*.coord[i][3,0])+*" "*

+str(*self*.coord[i][3,1])

)

if(roundness==0):

str1+=*" "*+*"0.1"*+*" "*+*"0.1"*+*" "*+*"0.1"*+*" "*+*"0.1"*

else:

str1+=(*" "*+str(roundness\**self*.cac[i][0])+*" "*+str(roundness\**self*.cac[i][1])+*" "*+str(roundness\**self*.cac[i][2])

+*" "*+str(roundness\**self*.cac[i][3])

)

fo.writelines(str1+*'\n'*)

fo.writelines(note+*'\n'*)

fo.close()

elif *self*.cb.currentText()==*'pentagon'*:

localtime=time.strftime(*"%Y-%m-%d %H.%M.%S"*,time.localtime())

fo=open(dir+*"/pentagon "*+localtime+*".txt"*,*"w"*)

roundness=float(*self*.LEc2.text())

fo.writelines(str(*self*.num)+*" "*+*"1"*+*"\n"*)

for i in range(0,*self*.x1.\_\_len\_\_()):

str1=(*"5"*+*" "*+str(*self*.coord[i][0,0])+*" "*+str(*self*.coord[i][0,1])+*" "*+str(*self*.coord[i][1,0])+*" "*+

str(*self*.coord[i][1,1])+*" "*+str(*self*.coord[i][2,0])+*" "*

+str(*self*.coord[i][2,1])+*" "*+str(*self*.coord[i][3,0])+*" "*+str(*self*.coord[i][3,1])+*" "*

+str(*self*.coord[i][4,0])+*" "*+str(*self*.coord[i][4,1])

)

if(roundness==0):

str1+=*" "*+*"0.1"*+*" "*+*"0.1"*+*" "*+*"0.1"*+*" "*+*"0.1"*+*" "*+*"0.1"*

else:

str1+=(*" "*+str(roundness\**self*.cac[i][0])+*" "*+str(roundness\**self*.cac[i][1])+*" "*+str(roundness\**self*.cac[i][2])

+*" "*+str(roundness\**self*.cac[i][3])+*" "*+str(roundness\**self*.cac[i][4])

)

fo.writelines(str1+*'\n'*)

fo.writelines(note+*'\n'*)

fo.close()

*self*.ifplot=0

def **comvisi**(*self*):

if *self*.cb.currentText()==*'ellipse'*:

*self*.Lbc.setText(*'L/D Ratio'*)

*self*.LEc.show()

*self*.Lbc.show()

elif (*self*.cb.currentText()==*'triangle'* or *self*.cb.currentText()==*'rectangle'* or *self*.cb.currentText()==*'pentagon'*):

*self*.Lbc.setText(*'Angular Offset（°）'*)

*self*.Lbc.show()

*self*.LEc.show()

*self*.Lbc2.show()

*self*.LEc2.show()

else:

*self*.Lbc.hide()

*self*.LEc.hide()

def **plot**(*self*):

x=[]

y=[]

if not *self*.infodic:

QMessageBox.warning(*self*, *'Warning'*, *'Please select an input file'*,QMessageBox.Yes)

return

for element in *self*.infodic:

x.append(log10(element))

y.append(*self*.infodic[element])

*self*.ax = *self*.fig.add\_subplot(111)

*self*.fig.subplots\_adjust(left=0.2,right=0.9,top=0.95,bottom=0.15)

*self*.ax.clear()

*self*.ax.hold(True)

#self.ax.set\_xscale('log')

max=int(x[0])+1

min=int(x[x.\_\_len\_\_()-1])-1

xlabels=[]

for i in range(min,max):

xlabels.append(10\*\*i)

xlabels.append(5\*10\*\*i)

xlabels.append(10\*\*max)

xticks=[]

for element in xlabels:

xticks.append(log10(element))

*self*.ax.xaxis.set\_major\_locator(ticker.FixedLocator(xticks))

*self*.ax.xaxis.set\_major\_formatter(ticker.FixedFormatter(xlabels))

*self*.ax.xaxis.grid(b=True)

*self*.ax.set\_xlabel(*u'Grain Diameter/mm'*,fontproperties=*'Times New Roman'*)

*self*.ax.set\_ylabel(*u'Percentage Finer (%)'*,fontproperties=*'Times New Roman'*)

*self*.ax.set\_xlim(xticks[0],xticks[xticks.\_\_len\_\_()-1])

*self*.ax.set\_ylim(0,100)

*self*.ax.grid(True)

x.append(xticks[0])

y.append(0)

line1, = *self*.ax.plot(x, y, *'-'*, linewidth=2, marker=*'^'*,

markersize=8, markerfacecolor=(0,0,0), label=*'Grading Curve'*)

*self*.ax.legend(loc=*'lower right'*)

xy=[]

for i in range(0,x.\_\_len\_\_()-1):

*self*.ax.text(x[i],y[i],str((xlabels[xlabels.\_\_len\_\_()-i-1],y[i])),fontsize=12)

# refresh canvas

*self*.canvas.draw()

def **plot2**(*self*):

if not *self*.infodic:

QMessageBox.warning(*self*, *'Warning'*, *'Please select an input file'*,QMessageBox.Yes)

return

projections.register\_projection(My\_Axes)

*self*.ax2 = *self*.fig2.add\_subplot(111,projection=*'My\_Axes'*)

*self*.ax2.clear()

*self*.ax2.hold(True)

m=*self*.cal()

N=m[0]

dmax=m[1]

w=int(m[2])

dic=m[3]

l=int(N/w)

N=int(w\*l)

*self*.num=N

d=[]

x=[]

y=[]

*"""*

*if(((w+1)/20\*0.9)>0.1):*

*self.fig2.subplots\_adjust(left=0.4\*(1-(w+1)/20\*0.9),right=0.6\*(1+(w+1)/20\*0.9),top=0.95,bottom=0.05)*

*elif(((w+1)/20\*0.9)>0.1):*

*self.fig2.subplots\_adjust(left=0,right=1,top=0.95,bottom=0.05)*

*else:*

*self.fig2.subplots\_adjust(left=0.4,right=0.6,top=0.95,bottom=0.05)*

*"""*

*self*.fig2.subplots\_adjust(left=0.1,right=0.9,top=0.95,bottom=0.05)

for D in dic.keys():

for j in range(0,dic[D]):

d.append(D)

x.append(random.uniform(D/2,2\*dmax-D/2))

y.append(random.uniform(D/2,2\*dmax-D/2))

if *self*.cb.currentText()==*'circle'*:

*self*.drawcircle(x, y, d, w, dmax, N, *'circle'*)

elif *self*.cb.currentText()==*'ellipse'*:

*self*.drawellipse(x, y, d, w, dmax, N)

elif *self*.cb.currentText()==*'triangle'*:

*self*.drawcircle(x, y, d, w, dmax, N, *'triangle'*)

elif *self*.cb.currentText()==*'rectangle'*:

*self*.drawcircle(x, y, d, w, dmax, N, *'rectangle'*)

elif *self*.cb.currentText()==*'pentagon'*:

*self*.drawcircle(x, y, d, w, dmax, N, *'pentagon'*)

xticks=[]

yticks=[]

for i in range(0,w+1):

xticks.append(2\*dmax\*i)

for i in range(0,l+1):

yticks.append(2\*dmax\*i)

xmajors=np.linspace(0,2\*dmax\*w,w+1)

*self*.ax2.xaxis.set\_major\_locator(ticker.FixedLocator(xmajors))

*self*.ax2.yaxis.set\_major\_locator(ticker.MaxNLocator(21))

yminors=np.linspace(0,2\*dmax\*l,l+1)

*self*.ax2.yaxis.set\_minor\_locator(ticker.FixedLocator(yminors))

for ymin in *self*.ax2.yaxis.get\_minorticklocs():

*self*.ax2.axhline(y=ymin, ls=*'--'*,lw=0.1,color=*'0.2'*)

*self*.ax2.xaxis.grid(b=True)

*self*.ax2.axis(*'equal'*)

#self.ax2.set\_aspect(1)

*self*.ax2.set\_xbound(lower=-dmax,upper=2\*dmax\*(w+0.5))

*self*.ax2.set\_ybound(lower=-dmax, upper=2\*dmax\*22.5)

*self*.canvas2.draw()

*self*.ifplot=1

def **drawcircle**(*self*,x,y,d,w,dmax,N,sh):

se=[]

*self*.x1.clear()

*self*.y1.clear()

*self*.dd.clear()

*self*.cac.clear()

*self*.coord.clear()

se=random.sample(range(0,N),N)

i=j=k=0

for n in se:

*self*.x1.append(x[n]+i\*2\*dmax)

*self*.y1.append(y[n]+j\*2\*dmax)

*self*.dd.append(d[n])

circle=mpatches.Circle((*self*.x1[k],*self*.y1[k]),d[n]/2)

if(sh==*'triangle'*):

circle.set\_facecolor(*'none'*)

circle.set\_edgecolor(*'b'*)

*self*.drawtriangle(*self*.x1[k], *self*.y1[k], d[n])

elif(sh==*'rectangle'*):

circle.set\_facecolor(*'none'*)

circle.set\_edgecolor(*'b'*)

*self*.drawrectangle(*self*.x1[k], *self*.y1[k], d[n])

elif(sh==*'pentagon'*):

circle.set\_facecolor(*'none'*)

circle.set\_edgecolor(*'b'*)

*self*.drawpentagon(*self*.x1[k], *self*.y1[k], d[n])

*self*.ax2.add\_patch(circle)

i+=1

k+=1

if i==w:

i=0

j+=1 # 0 to 15 point radii

def **drawellipse**(*self*,x,y,d,w,dmax,N):

se=[]

se=random.sample(range(0,N),N)

*self*.x1.clear()

*self*.y1.clear()

*self*.dd.clear()

*self*.an.clear() #重置输出项

i=j=k=0

ra=float(*self*.LEc.text())

for n in se:

*self*.x1.append(x[n]+i\*2\*dmax) #添加顺序

*self*.y1.append(y[n]+j\*2\*dmax)

*self*.an.append(random.uniform(0,180))

*self*.dd.append(d[n])

elli=mpatches.Ellipse((*self*.x1[k],*self*.y1[k]),d[n],d[n]/ra,*self*.an[k])

*self*.ax2.add\_patch(elli)

i+=1

k+=1

if i==w:

i=0

j+=1 # 0 to 15 point radii

def **drawtriangle**(*self*,x1,y1,d):

xy=[]

a=[0,120,240]

v=float(*self*.LEc.text())

va1=va2=0

va1=random.uniform(0,360)

for i in range(0,3):

va2=random.uniform(-1\*v,v)

a[i]+=va1+va2

for j in range(i,0,-1):

if(a[j]==a[j-1]):

a[j]+=1

for i in range(0,3):

for j in range(i+1,3):

if(a[i]>a[j]):

temp=a[j]

a[j]=a[i]

a[i]=temp

for i in range(0,3):

a[i]=np.deg2rad(a[i])

xy.append(x1+d/2\*np.cos(a[i]))

xy.append(y1+d/2\*np.sin(a[i]))

Path = mpath.Path

path\_data = [

(Path.MOVETO, (xy[0],xy[1])),

(Path.LINETO, (xy[2],xy[3])),

(Path.LINETO, (xy[4],xy[5])),

(Path.CLOSEPOLY, (xy[0],xy[1])),

]

codes, verts = zip(\*path\_data)

path = mpath.Path(verts, codes)

patch = mpatches.PathPatch(path, facecolor=*'r'*, alpha=0.5)

*self*.ax2.add\_patch(patch)

*self*.coord.append(np.mat([[xy[0]/1000,xy[1]/1000],[xy[2]/1000,xy[3]/1000],[xy[4]/1000,xy[5]/1000]]))

cac1=[]

for i in range(0,2):

cac1.append(a[i+1]-a[i])

cac1.append((a[0]+2\*pi)-a[2])

*self*.cac.append([cac1[0],cac1[1],cac1[2]])

def **drawrectangle**(*self*,x1,y1,d):

a=[0,90,180,270]

v=float(*self*.LEc.text())

va1=va2=0

va1=random.uniform(0,360)

for i in range(0,4):

va2=random.uniform(-1\*v,v)

a[i]+=va1+va2

for i in range(0,4): #各点逆时针排序

for j in range(i+1,4):

if(a[i]>a[j]):

temp=a[j]

a[j]=a[i]

a[i]=temp

xy=[]

for i in range(0,4):

a[i]=np.deg2rad(a[i])

xy.append(x1+d/2\*np.cos(a[i]))

xy.append(y1+d/2\*np.sin(a[i]))

Path = mpath.Path

path\_data = [

(Path.MOVETO, (xy[0],xy[1])),

(Path.LINETO, (xy[2],xy[3])),

(Path.LINETO, (xy[4],xy[5])),

(Path.LINETO, (xy[6],xy[7])),

(Path.CLOSEPOLY, (xy[0],xy[1])),

]

codes, verts = zip(\*path\_data)

path = mpath.Path(verts, codes)

patch = mpatches.PathPatch(path, facecolor=*'b'*, alpha=0.5)

*self*.ax2.add\_patch(patch)

*self*.coord.append(np.mat([[xy[0]/1000,xy[1]/1000],[xy[2]/1000,xy[3]/1000],[xy[4]/1000,xy[5]/1000],

[xy[6]/1000,xy[7]/1000]])) #添加坐标点及圆心角数据

cac1=[]

for i in range(0,3):

cac1.append(a[i+1]-a[i])

cac1.append((a[0]+2\*pi)-a[3])

*self*.cac.append([cac1[0],cac1[1],cac1[2],cac1[3]])

def **drawpentagon**(*self*,x1,y1,d):

a=[0,72,144,216,288]

v=float(*self*.LEc.text())

va1=va2=0

va1=random.uniform(0,360) #三点重合可能性

for i in range(0,5):

va2=random.uniform(-1\*v,v)

a[i]+=va1+va2

for i in range(0,5):

for j in range(i+1,5):

if(a[i]>a[j]):

temp=a[j]

a[j]=a[i]

a[i]=temp

xy=[]

for i in range(0,5):

a[i]=np.deg2rad(a[i])

xy.append(x1+d/2\*np.cos(a[i]))

xy.append(y1+d/2\*np.sin(a[i]))

Path = mpath.Path

path\_data = [

(Path.MOVETO, (xy[0],xy[1])),

(Path.LINETO, (xy[2],xy[3])),

(Path.LINETO, (xy[4],xy[5])),

(Path.LINETO, (xy[6],xy[7])),

(Path.LINETO, (xy[8],xy[9])),

(Path.CLOSEPOLY, (xy[0],xy[1])),

]

codes, verts = zip(\*path\_data)

path = mpath.Path(verts, codes)

patch = mpatches.PathPatch(path, facecolor=*'b'*, alpha=0.5)

*self*.ax2.add\_patch(patch)

*self*.coord.append(np.mat([[xy[0]/1000,xy[1]/1000],[xy[2]/1000,xy[3]/1000],[xy[4]/1000,xy[5]/1000],[xy[6]/1000,xy[7]/1000],

[xy[8]/1000,xy[9]/1000]]))

cac1=[]

for i in range(0,4):

cac1.append(a[i+1]-a[i])

cac1.append((a[0]+2\*pi)-a[4])

*self*.cac.append([cac1[0],cac1[1],cac1[2],cac1[3],cac1[4]])

def **calelli**(*self*,x,y,a,b,an):

*self*.cac=[]

an2=np.deg2rad(an)

trans=np.mat([[np.cos(an2),-np.sin(an2)],[np.sin(an2),np.cos(an2)]])

#备选计算公式

#c=sqrt(a\*\*2+b\*\*2)

#d=a-b

#cac1=np.arcsin(((a\*\*2-b\*\*2)+(c-b)\*d+b\*c)/(2\*a\*(b+(a\*\*2-b\*\*2)\*c+(a\*\*2+b\*\*2)\*d)/b/c))

#cac2=np.arcsin((c-d)/2/(a-((a\*\*2-b\*\*2)\*c+(a\*\*2+b\*\*2)\*d)/a/c))

h=(a-b)\*(a+b+sqrt(a\*\*2+6\*a\*b+b\*\*2))/(a-b+sqrt(a\*\*2+6\*a\*b+b\*\*2))

k=(a-b)\*(a+3\*b+sqrt(a\*\*2+6\*a\*b+b\*\*2))/(4\*b)

xj=h\*((a-h)/sqrt(k\*\*2+h\*\*2)+1)

yj=k\*(a-h)/sqrt(k\*\*2+h\*\*2)

xy1=np.mat([xj,yj])

xy1=xy1\*trans

xy1=np.mat([xy1[0,0]+x,xy1[0,1]+y])

xy2=np.mat([-xj,yj])

xy2=xy2\*trans

xy2=np.mat([xy2[0,0]+x,xy2[0,1]+y])

xy3=np.mat([-xj,-yj])

xy3=xy3\*trans

xy3=np.mat([xy3[0,0]+x,xy3[0,1]+y])

xy4=np.mat([xj,-yj])

xy4=xy4\*trans

xy4=np.mat([xy4[0,0]+x,xy4[0,1]+y])

cac1=2\*np.arctan(yj/(xj-h))

cac2=pi-cac1

*self*.cac=[cac2,cac1,cac2,cac1]

return(xy1/1000,xy2/1000,xy3/1000,xy4/1000)

def **cal**(*self*):

N=int(*self*.LEn.text())

a=[]

d=[]

p=[]

for element in *self*.infodic:

d.append(element)

p.append(*self*.infodic[element])

for i in range(0,p.\_\_len\_\_()):

p[i]=float(p[i])

#确定计算粒径范围

ma=0

mi=p.\_\_len\_\_()-1

for i in range(0,p.\_\_len\_\_()):

if p[i]==100:

continue

else:

dmax=d[i-1]

ma=i-1

break

for i in range(1,p.\_\_len\_\_()):

if dmax/d[i]<=10:

continue

else:

dmin=d[i-1]

mi=i-1

break

for i in range(mi+1,p.\_\_len\_\_()):

p[i]=0

for i in range(0,p.\_\_len\_\_()-1):

temp=[]

for j in range(0,p.\_\_len\_\_()):

if(j==i):

temp.append(1-0.01\*(p[i]-p[i+1]))

else:

temp.append(-0.01\*(p[i]-p[i+1])\*(d[j]\*\*2)/(d[i]\*\*2))

a.append(temp)

temp=[]

for i in range(0,p.\_\_len\_\_()):

temp.append(1)

a.append(temp)

a=np.array(a)

b=[]

for i in range(0,p.\_\_len\_\_()-1):

b.append(0)

b.append(1)

b=np.array(b)

x=np.linalg.solve(a,b)

c=[]

for i in range(0,p.\_\_len\_\_()):

t=int(round(x[i]\*N))

c.append(t)

for i in range(0,p.\_\_len\_\_())[::-1]: #确定计算中最小粒径

if c[i]!=0:

break

s=0

for j in range(0,p.\_\_len\_\_()):

s+=c[j]

c[i]+=N-s

dic={}

for i in range(ma,mi+1):

dic[d[i]]=c[i]

vd=0

for element in dic:

vd+=dic[element]\*pi\*(element\*\*2)/4

v=(1+float(*self*.LEe.text()))\*vd

ra=float(*self*.LEr.text())

wid=sqrt(v/ra)

w=round(wid/(2\*dmax))

return N,dmax,w,dic

if \_\_name\_\_==*'\_\_main\_\_'*:

app=0

app=QApplication(sys.argv)

times=QFont(*'Times New Roman'*,10)

app.setFont(times)

m=Window()

m.show()

sys.exit(app.exec\_())