|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

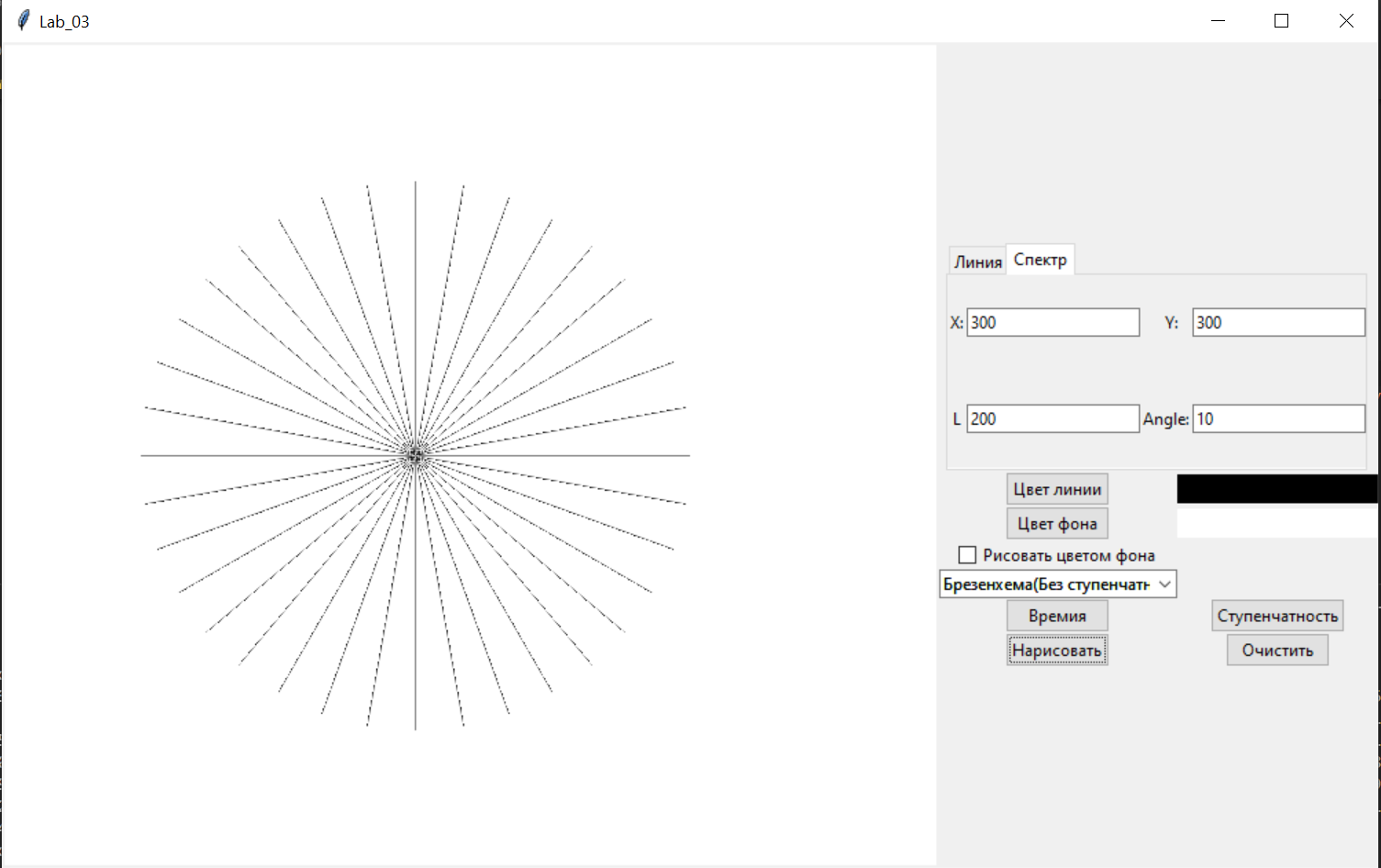
**Лабораторная работа № 3**

|  |  |
| --- | --- |
| **Тема: Реализация и исследование алгоритмов построения отрезков**  **Студент: Нгуен Ань Тхы**  **Группа: ИУ7-46Б**  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель: Куров.А.В** |  |

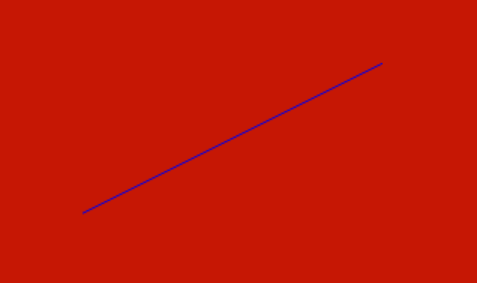
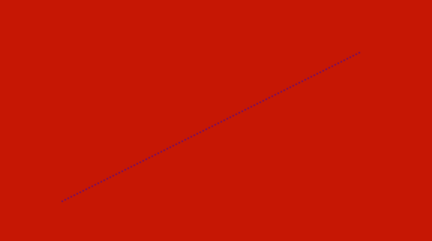
Москва.

2020 г.

**Интерфей:**

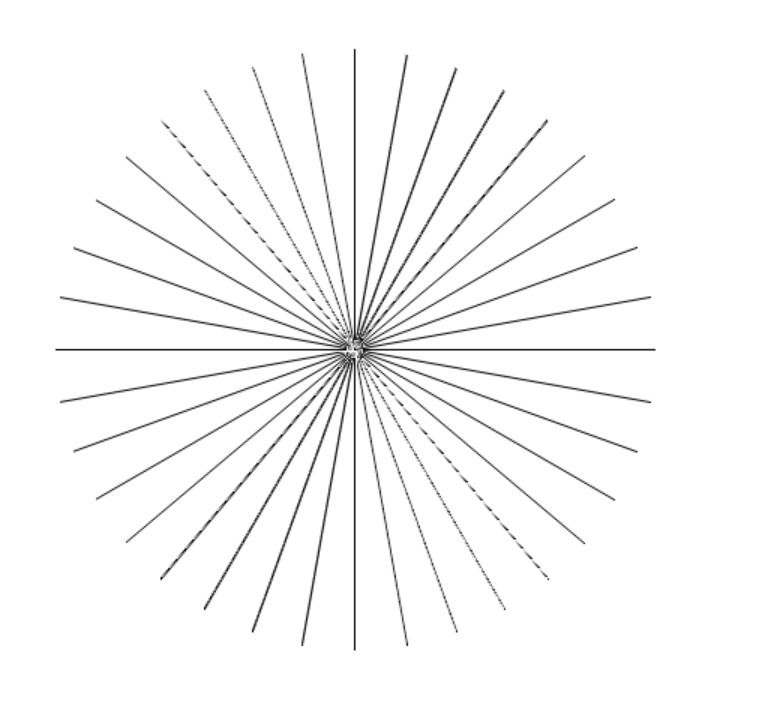
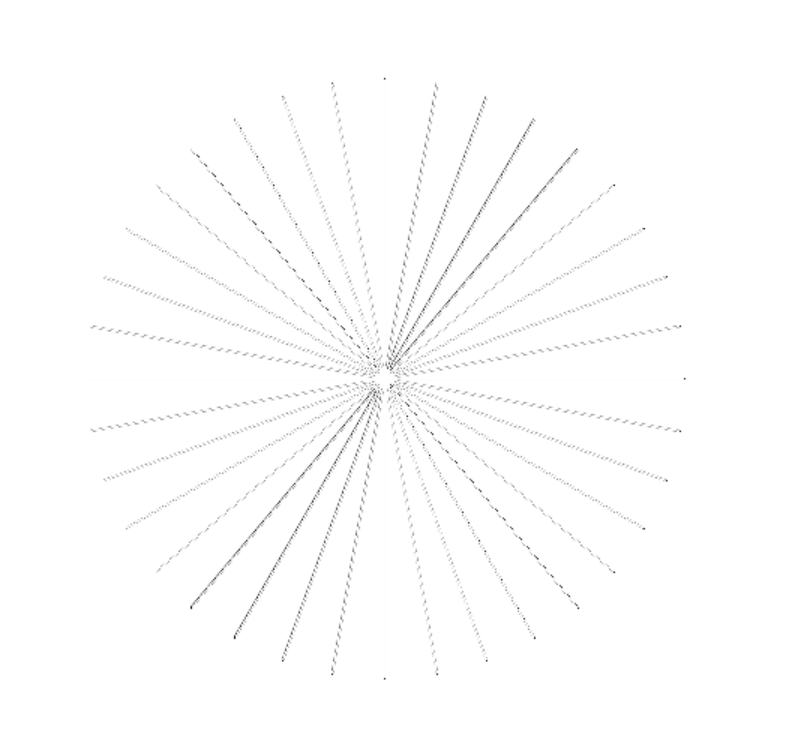
****

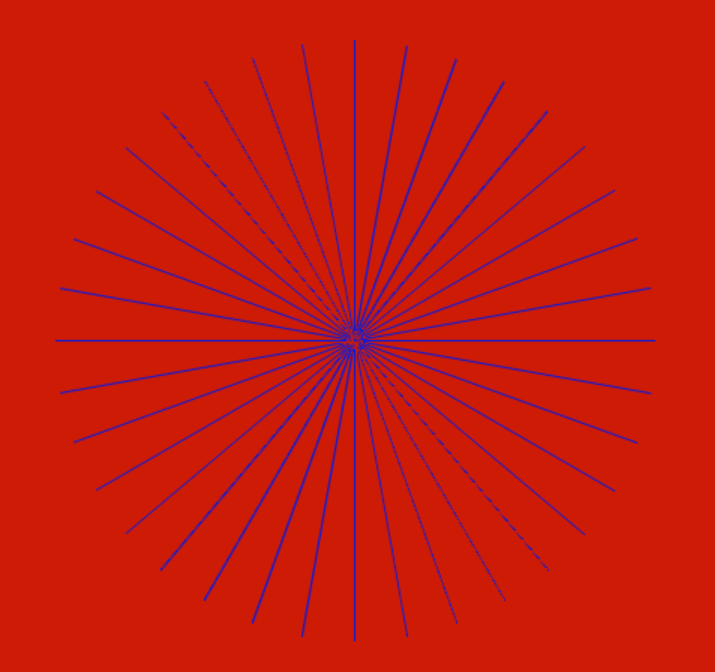
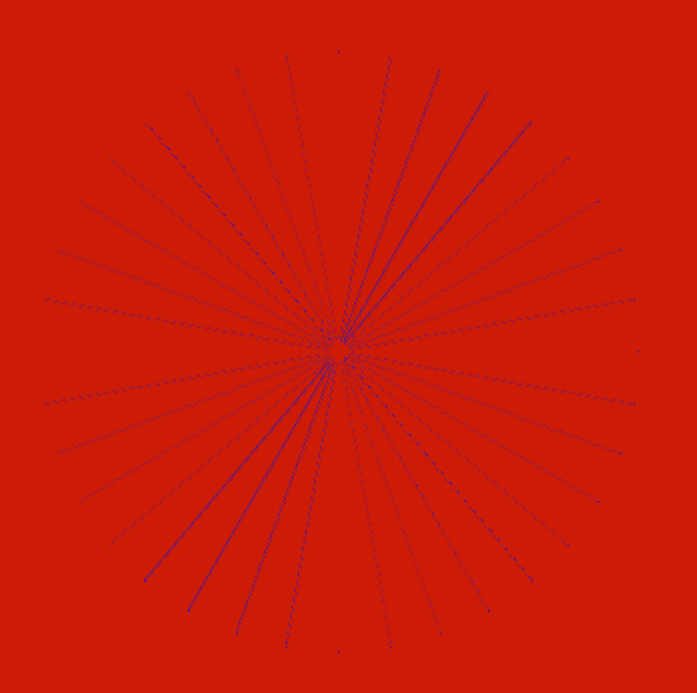
1. **Рисование отдельных отрезков и сравнение их визуальных характеров:**

**** ****

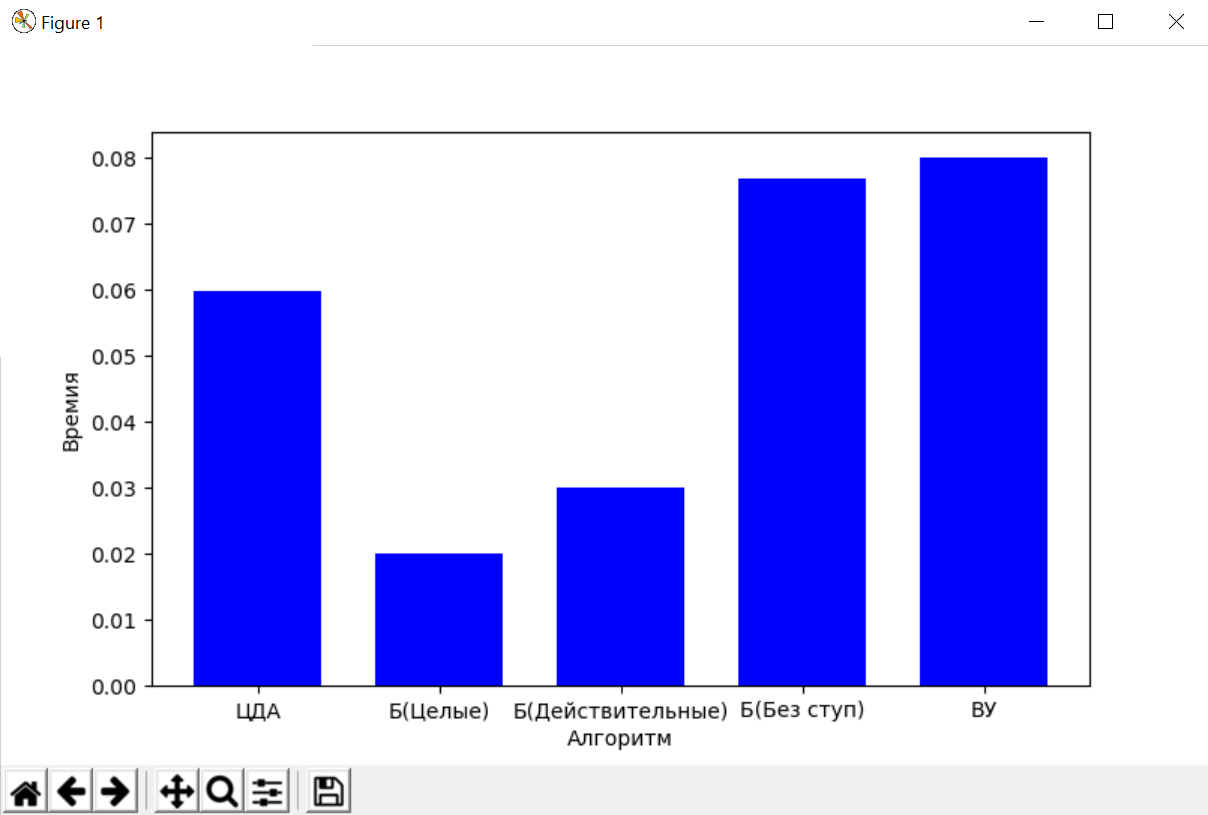
1. **Исследование визуальных характеристик отрезков во всем спектре углов:**

Длина дуги: 200, угла: 10o, Рисование цвотом фона

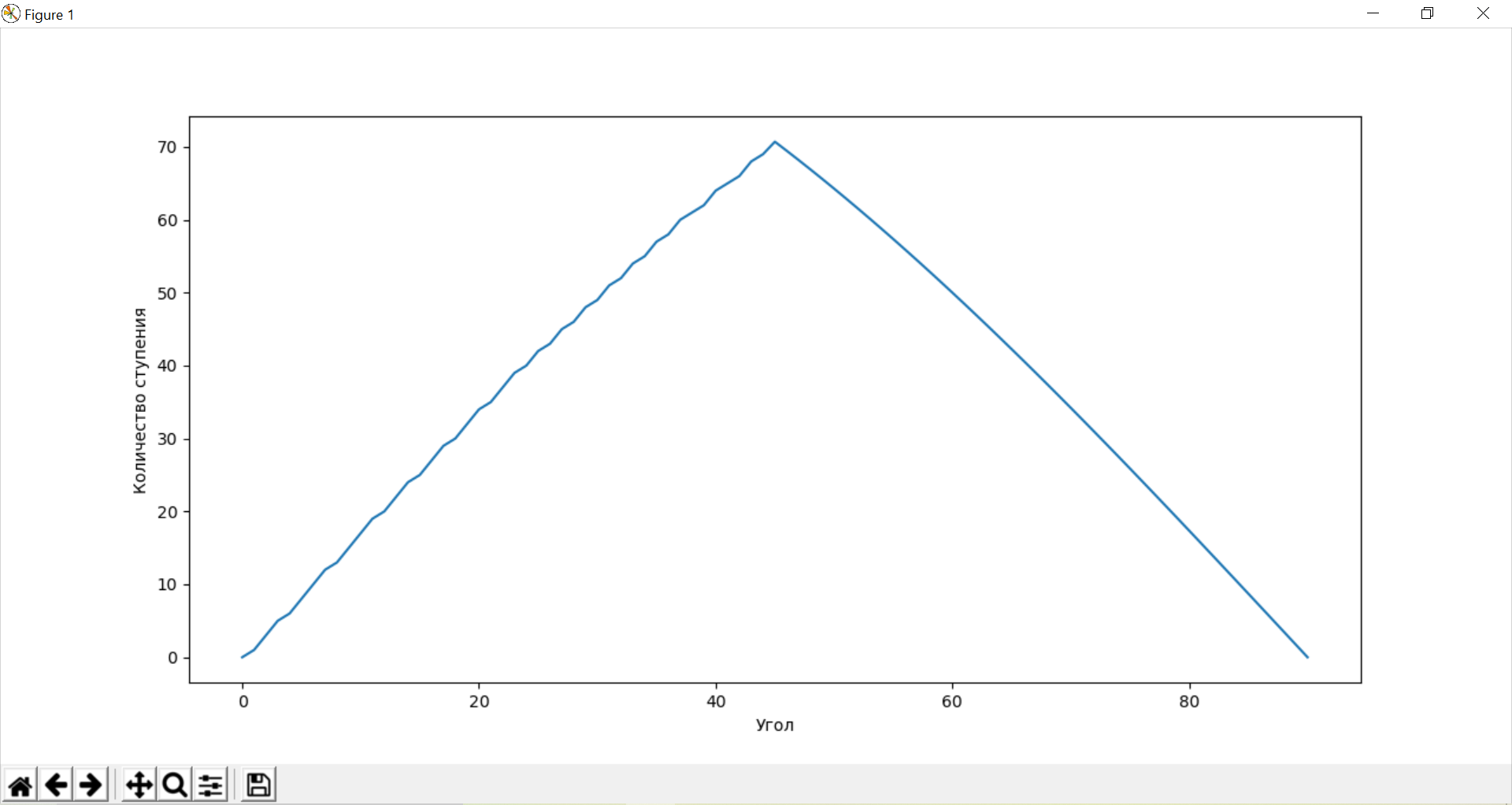
 ****

**** ****

1. **Исследование временых характеристик:**

****

1. **Исследование ступенчастности:**

****

1. **Алгоритм:**
2. **Цифровой дифференциальный анализатор (ЦДА):**

1. (xн , yн), (xк , yк)

2. Δx = xк - xн, Δy = yк - yн

3. dx = | Δx|, dy = | Δy|

4. если dx > dy, то l = dx иначе l = dy

5. Δx = Δx / l, Δy = Δy / l

6. x = xн , y = yн

7. цикл построения отрезка (по i-ой до l+1)

7.1 высвечивание точки (E(x), E(y))

7.2 x = x + Δx, y = y + Δy

8. конец

1. **Брезенхема с действительными числами**

Алгоритм:

1. ввод (xн , yн), (xк , yк)

2. x = xн, y = yк

3. dx = xк - xн , dy = yк - yн

Δx = sign(dx) Δy = sign(dy), где sign(x) =

4. dx = | dx|, dy = | dy |

5. Если dx = dy, то change = 0,

иначе change = 1, t = dx, dx = dy, dy = t

6. tg = dy / dx, e = tg – 1/2

7. Цикл построения (i от 1 до dx+1)

7.1 высвечивание точку (x, y)

7.2 Если e > 0, то

а) Если change = 0 то y = y + Δy иначе x = x + Δx

б) e = e – 1

Иначе:

а) Если change = 0, то x = x + Δx иначе y = y + Δy

б) e = e + tg

8. Конец

1. **Брезенхема с целыми числами**

1. ввод (xн , yн), (xк , yк)

2. x = xн, y = yк

3. dx = xк - xн , dy = yк - yн

Δx = sign(dx) Δy = sign(dy), где sign(x) =

4. dx = | dx|, dy = | dy |

5. Если dx = dy, то change = 0,

иначе change = 1, t = dx, dx = dy, dy = t

6. у = 2 \* dy - dx

7. Цикл построения (i от 1 до dx+1)

7.1 высвечивание точку (x, y)

7.2 Если e > 0, то

а) Если change = 0 то y = y + Δy иначе x = x + Δx

б) e = e – 2

Иначе:

а) Если change = 0, то x = x + Δx иначе y = y + Δy

б) e = e + 2 \* dy

8. Конец

1. **Брезенхема с устранением ступенчастности**
2. Ввод (XН,YН), (XК,YК) , I - количество уровней интенсивности
3. Проверка вырожденности отрезка. Если отрезок вырожден, то высвечивание отдельного пиксела и переход к п.13
4. dX=Xк-Xн и dY=Yк-Yн
5. SX = sign(dX), SY=sign(dY)
6. dX=|dX|, dY=|dY|
7. m = dY / dX (тангенса угла)
   1. если m > 1, то t = dX; dX = dY; dY = t; m = 1 / m; fl = 1
   2. если m < 1, то fl = 0
8. ошибки e = I / 2
9. X = Xн, Y = Yн
10. m = m \* I (тангенса угла) и коэффициента W = I - m
11. Высвечивание точку (X,Y) интенсивностью I(e)
12. Цикл от i=1 до dX
    1. Если e < W, то  
       a) если fl = 0, то X = X + SX  
       b) если fl = 1, то Y = Y + SY  
       c) e = e + m
    2. Если e > W, то X = X + SX, Y = Y + SY, e = e-W
    3. Высвечивание точку (X,Y) интенсивностью I(e)
13. Конец
14. **Алгоритм Ву**

1. Ввод (XH,YH), (XK,YK) , I - количество уровней интенсивности

2. dx = xK – xH, dy = xK – yH

3. Если abs(x) < abs(dy), то обмен x1, y1; x2, y2; dx, dy

4. Если x2 < x1, то обмен x1, x2; y1, y2

5. m = dy / dx (тангенса угла)

6. x\_end = round(xH), y\_end = yH + m \* (x\_end – xH)

X1 = x\_end

Y = y\_end + m

x\_end = round(xK + 1/2)

X2 = x\_end

(X2, Y2) : конечная точка в цикле

7. Цикл построения (X от X1 до X2)

7.1 Высвечивание точку (X, ipart(Y), 1 - fpart(Y))

7.2 Высвечивание точку (X, ipart(Y) + 1, fpart(Y))

{ipart(Y): целая часть от Y, fpart(Y): дробная часть от Y}

7.3 Y = Y + m

**VI. Код программы:**

from math import \*

from PyQt5.QtGui import QColor

width\_line = 1 #constant

def draw\_pix(self, x, y, color):

self.canvas.create\_line(x, y, x + 1, y + 1,fill = color, width = width\_line)

return 0

def sign(x):

if x > 0:

return 1

elif x < 0:

return -1

else:

return 0

def fpart(x):

return x - int(x)

def rpart(x):

return 1 - fpart(x)

#ЦДА

def drawline\_dda(self, x1, y1, x2, y2):

# my code

lenx = int(x2 - x1)

leny = int(y2 - y1)

dx = abs(lenx)

dy = abs(leny)

step = max(dx, dy)

x\_inc = lenx / step

y\_inc = leny / step

x, y = x1, y1

for i in range(step + 1):

draw\_pix(self, round(x), round(y), self.color\_pen)

x += x\_inc

y += y\_inc

return 0

#

def drawline\_br\_float(self, x1, y1, x2, y2):

dx = x2 - x1

dy = y2 - y1

if dx == 0 and dy == 0:

draw\_pix(self, x1, x1, self.color\_pen)

return 1

x = x1

y = y1

sx = sign(dx)

sy = sign(dy)

dx = abs(dx)

dy = abs(dy)

if (dx > dy):

change = 0

else:

change = 1

dx, dy = dy, dx

m = dy / dx

e = m - 0.5

while (x != x2) or (y != y2):

draw\_pix(self, x, y, self.color\_pen)

if e >= 0:

if change == 0:

y += sy

else:

x += sx

e -= 1

if e < 0:

if change == 0:

x += sx

else:

y += sy

e += m

return 0

def drawline\_br\_int(self, x1, y1, x2, y2):

dx = x2 - x1

dy = y2 - y1

if dx == 0 and dy == 0:

draw\_pix(self, x1, y1, self.color\_pen)

return 1

sx = sign(dx)

sy = sign(dy)

dx = abs(dx)

dy = abs(dy)

if (dx > dy):

change = 0

else:

change = 1

dx, dy = dy, dx

e = 2 \* dy - dx

x, y = x1, y1

while (x != x2) or (y != y2):

draw\_pix(self, x, y, self.color\_pen)

if e >= 0:

if change == 0:

y += sy

else:

x += sx

e -= 2 \* dx

else:

if change == 0:

x += sx

else:

y += sy

e += 2 \* dy

return 0

def drawline\_br\_smooth(self, x1, y1, x2, y2, fill, I):

dx = x2 - x1

dy = y2 - y1

if dx == 0 and dy == 0:

draw\_pix(self, x1, y1, self.color\_pen)

return 1

sx = sign(dx)

sy = sign(dy)

dx = abs(dx)

dy = abs(dy)

if (dy > dx):

dx, dy = dy, dx

change = 1

else:

change = 0

try:

tg = dy / dx \* I

except ZeroDivisionError:

tg = 1

e = I / 2

w = I - tg

x = x1

y = y1

while (x != x2) or (y != y2):

draw\_pix(self, x, y, fill[round(e) - 1])

if e >= w:

y += sy

x += sx

e -= w

else:

if change == 0:

x += sx

else:

y += sy

e += tg

return 0

def drawline\_wu(self, x1, y1, x2, y2, fill, I):

if x1 == x2 and y1 == y2:

draw\_pix(self, x1, y1, self.color\_pen)

steep = abs(y2 - y1) > abs(x2 - x1)

if steep:

x1, y1 = y1, x1

x2, y2 = y2, x2

if x1 > x2:

x1, x2 = x2, x1

y1, y2 = y2, y1

dx = x2 - x1

dy = y2 - y1

if dx == 0:

tg = 1

else:

tg = dy / dx

#first end point

xend = round(x1)

yend = y1 + tg \* (xend - x1)

xpx1 = xend

y = yend + tg

#second end point

xend = int(x2 + 0.5)

xpx2 = xend

if steep:

for x in range(xpx1, xpx2):

self.canvas.create\_line(int(y), x, int(y) + 1, int(x) + 2, fill = fill[int((I - 1) \* abs(1 - y + int(y)))])

self.canvas.create\_line(int(y) + 1, int(x) + 1, int(y) + 2, int(x) + 2, fill = fill[int((I - 1) \* abs(y - int(y)))])

y += tg

else:

for x in range(xpx1, xpx2):

self.canvas.create\_line(int(x) + 1, int(y), int(x) + 2, int(y) + 1, fill = fill[round((I - 1) \* abs(1 - y + floor(y)))])

self.canvas.create\_line(int(x) + 1, int(y) + 1, int(x) + 2, int(y) + 2, fill = fill[round((I - 1) \* abs(y - floor(y)))])

y += tg

def get\_rgb\_intensity(self, line\_color, bg\_color, intensity):

grad = []

(r1, g1, b1) = self.winfo\_rgb(line\_color) #get tuple of color of line

(r2, g2, b2) = self.winfo\_rgb(bg\_color) #get tuple of color of background

r\_ratio = float(r2 - r1) / intensity

g\_ratio = float(g2 - g1) / intensity

b\_ratio = float(b2 - b1) / intensity

for i in range(intensity):

nr = int(r1 + r\_ratio \* i)

ng = int(g1 + g\_ratio \* i)

nb = int(b1 + b\_ratio \* i)

grad.append("#%4.4x%4.4x%4.4x" % (nr, ng, nb))

return grad

def drawline\_lib(self, x1, y1, x2, y2):

self.canvas.create\_line(x1, y1, x2 + 1, y2 + 1, fill = self.color\_pen, width = width\_line)

return 0

from math import \*

from time import time

from PyQt5.QtGui import QColor

from matplotlib import pyplot as plt

width\_line = 1 #constant

def draw\_pix(self, x, y, color):

self.canvas.create\_line(x, y, x + 1, y + 1,fill = color, width = width\_line)

# def draw\_pix(self, x1, y1, x2, y2, color):

# self.canvas.creat\_line(x1, y1, x2, y2, fill = color)

def sign(x):

if x > 0:

return 1

elif x < 0:

return -1

else:

return 0

#ЦДА

def time\_DDA(x1, y1, x2, y2):

# points = []

time\_beg = time()

lenx = (x2 - x1)

leny = (y2 - y1)

dx = abs(lenx)

dy = abs(leny)

step = max(dx, dy)

x\_inc = lenx / step

y\_inc = leny / step

x, y = x1, y1

for i in range(step + 1):

# draw\_pix(self, int(x), int(y), self.color\_pen)

# points.append([x, y, self.color\_pen])

x += x\_inc

y += y\_inc

# return points

time\_end = time()

return time\_end - time\_beg

#

def time\_br\_float(x1, y1, x2, y2):

# points = []

time\_beg = time()

dx = x2 - x1

dy = y2 - y1

if dx == 0 and dy == 0:

# draw\_pix(self, x1, x1, self.color\_pen)

# points.append([x1, y1, self.color\_pen])

return time() - time\_beg

x = x1

y = y1

sx = sign(dx)

sy = sign(dy)

dx = abs(dx)

dy = abs(dy)

if (dx > dy):

change = 0

else:

change = 1

dx, dy = dy, dx

m = dy / dx

e = m - 0.5

while (x != x2) or (y != y2):

# draw\_pix(self, x, y, self.color\_pen)

# points.append([x, y, self.color\_pen])

if e >= 0:

if change == 0:

y += sy

else:

x += sx

e -= 1

if e < 0:

if change == 0:

x += sx

else:

y += sy

e += m

# return points

time\_end = time()

return time\_end - time\_beg

def time\_br\_int(x1, y1, x2, y2):

# points = []

time\_beg = time()

dx = x2 - x1

dy = y2 - y1

if dx == 0 and dy == 0:

# draw\_pix(self, x1, y1, self.color\_pen)

# points.append([x1, y1, self.color\_pen])

return points

sx = sign(dx)

sy = sign(dy)

dx = abs(dx)

dy = abs(dy)

if (dx > dy):

change = 0

else:

change = 1

dx, dy = dy, dx

e = 2 \* dy - dx

x, y = x1, y1

while (x != x2) or (y != y2):

# draw\_pix(self, x, y, self.color\_pen)

# points.append([x, y, self.color\_pen])

if e >= 0:

if change == 0:

y += sy

else:

x += sx

e -= 2 \* dx

else:

if change == 0:

x += sx

else:

y += sy

e += 2 \* dy

# return points

time\_end = time()

return time\_end - time\_beg

def time\_br\_smooth(x1, y1, x2, y2, fill, I):

# points = []

time\_beg = time()

grad = get\_intensity("#000000", "#ffffff", 50000)

dx = x2 - x1

dy = y2 - y1

if dx == 0 and dy == 0:

# draw\_pix(self, x1, y1, self.color\_pen)

# points.append([x1, y1, self.color\_pen])

return time() - time\_beg

sx = sign(dx)

sy = sign(dy)

dx = abs(dx)

dy = abs(dy)

if (dy > dx):

dx, dy = dy, dx

change = 1

else:

change = 0

try:

tg = dy / dx \* I

except ZeroDivisionError:

tg = 1

e = I / 2

w = I - tg

x = x1

y = y1

while (x != x2) or (y != y2):

# draw\_pix(self, x, y, fill[round(e) - 1])

# points.append([x, y, self.color\_pen])

if e >= w:

y += sy

x += sx

e -= w

else:

if change == 0:

x += sx

else:

y += sy

e += tg

# return points

time\_end = time()

return time\_end - time\_beg

def time\_wu(x1, y1, x2, y2, fill, I):

# points = []

time\_beg = time()

grad = get\_intensity("#000000", "#ffffff", 70000)

if x1 == x2 and y1 == y2:

# draw\_pix(self, x1, y1, self.color\_pen)

# points.append([x1, y1, self.color\_pen])

return points

steep = abs(y2 - y1) > abs(x2 - x1)

if steep:

x1, y1 = y1, x1

x2, y2 = y2, x2

if x1 > x2:

x1, x2 = x2, x1

y1, y2 = y2, y1

dx = x2 - x1

dy = y2 - y1

if dx == 0:

tg = 1

else:

tg = dy / dx

#first end point

xend = round(x1)

yend = y1 + tg \* (xend - x1)

xpx1 = xend

y = yend + tg

#second end point

xend = int(x2 + 0.5)

xpx2 = xend

if steep:

for x in range(xpx1, xpx2):

# self.canvas.create\_line(int(y), x, int(y) + 1, int(x) + 2, fill = fill[int((I - 1) \* abs(1 - y + int(y)))])

# self.canvas.create\_line(int(y) + 1, int(x) + 1, int(y) + 2, int(x) + 2, fill = fill[int((I - 1) \* abs(y - int(y)))])

# points.append([int(y), x, int(y) + 1, int(x) + 2, fill[int((I - 1) \* abs(1 - y + int(y)))]])

# points.append([int(y) + 1, int(x) + 1, int(y) + 2, int(x) + 2, fill[int((I - 1) \* abs(y - int(y)))]])

y += tg

else:

for x in range(xpx1, xpx2):

# self.canvas.create\_line(int(x) + 1, int(y), int(x) + 2, int(y) + 1, fill = fill[round((I - 1) \* abs(1 - y + floor(y)))])

# self.canvas.create\_line(int(x) + 1, int(y) + 1, int(x) + 2, int(y) + 2, fill = fill[round((I - 1) \* abs(y - floor(y)))])

# points.append([int(x) + 1, int(y), int(x) + 2, int(y) + 1, fill[round((I - 1) \* abs(1 - y + floor(y)))]])

# points.append([int(x) + 1, int(y) + 1, int(x) + 2, int(y) + 2, fill[round((I - 1) \* abs(y - floor(y)))]])

y += tg

# return points

time\_end = time()

return time\_end - time\_beg

def get\_intensity(line\_color, bg\_color, intensity):

grad = []

# (r1, g1, b1) = self.winfo\_rgb(line\_color) #get tuple of color of line

# (r2, g2, b2) = self.winfo\_rgb(bg\_color) #get tuple of color of background

r1, g1, b1 = 0, 0 ,0

r2, g2, b2 = 50, 50, 50

r\_ratio = float(r2 - r1) / intensity

g\_ratio = float(g2 - g1) / intensity

b\_ratio = float(b2 - b1) / intensity

for i in range(intensity):

nr = int(r1 + r\_ratio \* i)

ng = int(g1 + g\_ratio \* i)

nb = int(b1 + b\_ratio \* i)

grad.append("#%4.4x%4.4x%4.4x" % (nr, ng, nb))

# grad.reverse()

return grad

def time\_test(N):

res = []

x1 = 0

y1 = 0

x2 = x1 + N

y2 = y1 + N

# grad = get\_rgb\_intensity(self, self.color\_line, self.color\_bg, 100)

grad = ["#fff" for i in range(100)]

res.append(time\_DDA(x1, y1, x2, y2))

res.append(time\_br\_float(x1, y1, x2, y2))

res.append(time\_br\_int(x1, y1, x2, y2))

res.append(time\_br\_smooth(x1, y1, x2, y2, grad, 100))

res.append(time\_wu(x1, y1, x2, y2, grad, 100))

name = ["ЦДА", "Б(Целые)", "Б(Действительные)", "Б(Без ступ)", "ВУ"]

plt.xlabel("Алгоритм")

plt.ylabel("Времия")

plt.bar(name, res, width = 0.7, color = 'blue')

plt.show()

return res

def stepping():

x\_axis = []

y\_axis = []

for i in range(91):

x\_axis.append(i)

if i < 45:

y\_axis.append(floor(100 \* sin(i \* pi / 180)))

else:

y\_axis.append(floor(100) \* cos(i \* pi / 180))

plt.plot(x\_axis, y\_axis)

plt.xlabel("Угол")

plt.ylabel("Количество ступения")

plt.show()

import tkinter as tk

import tkinter.ttk as ttk

from tkinter import messagebox

from tkinter.colorchooser import askcolor

from matplotlib import pyplot as plt

from algorithms import \*

from time\_testing import \*

from typing import NamedTuple

class Line(NamedTuple):

x1: int

y1: int

x2: int

y2: int

color : str

alg: int

class Spectrum(NamedTuple):

x: int

y: int

r: int

angle: int

color: str

alg: int

# App class

class Kg4App(tk.Tk):

def \_\_init\_\_(self, \*args, \*\*kwargs):

tk.Tk.\_\_init\_\_(self)

options\_size = 336 # const

can\_x = args[0] - options\_size

can\_y = args[1]

tk.Tk.title(self, "Lab\_03")

tk.Tk.geometry(self, str(can\_x + options\_size) + "x" + str(can\_y))

container = tk.Frame(self)

container.pack(side="top", fill="both", expand=True)

container.grid\_rowconfigure(0, weight=1)

container.grid\_columnconfigure(0, weight=1)

self.frames = {}

for F in (MainPage, GraphPage):

frame = F(container, self)

self.frames[F] = frame

frame.grid(row=0, column=0, sticky="nsew")

self.show\_frame(MainPage)

def show\_frame(self, cont):

frame = self.frames[cont]

frame.tkraise()

class MainPage(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self, parent)

self.color\_bg = "#fff"

self.color\_line = "#000000"

self.color\_pen = self.color\_line

self.objects = []

self.res = []

# TABS

self.work = tk.Frame(self)

self.tab\_parent = ttk.Notebook(self.work, height=140)

self.tab1 = ttk.Frame(self.tab\_parent)

self.tab2 = ttk.Frame(self.tab\_parent)

self.tab\_parent.add(self.tab1, text="Линия")

self.tab\_parent.add(self.tab2, text="Спектр")

self.tab\_parent.grid(row=0, column=0, columnspan=2)

#Variable Tab1

self.var1\_1 = tk.StringVar()

self.var2\_1 = tk.StringVar()

self.var3\_1 = tk.StringVar()

self.var4\_1 = tk.StringVar()

#Variable Tab2

self.var1\_2 = tk.StringVar()

self.var2\_2 = tk.StringVar()

self.var3\_2 = tk.StringVar()

self.var4\_2 = tk.StringVar()

# Tab 1

self.e\_w = 20

self.tab1\_label\_1 = ttk.Label(self.tab1, text="X1:").grid(row=0, column=0)

self.tab1\_label\_2 = ttk.Label(self.tab1, text=" Y1:").grid(row=0, column=2)

self.tab1\_label\_3 = ttk.Label(self.tab1, text="X2:").grid(row=1, column=0)

self.tab1\_label\_4 = ttk.Label(self.tab1, text=" Y2:").grid(row=1, column=2)

self.tab1\_entry\_x1 = ttk.Entry(self.tab1, textvariable=self.var1\_1, width=self.e\_w).grid(row=0, column=1)

self.tab1\_entry\_y1 = ttk.Entry(self.tab1, textvariable=self.var2\_1, width=self.e\_w).grid(row=0, column=3)

self.tab1\_entry\_x2 = ttk.Entry(self.tab1, textvariable=self.var3\_1, width=self.e\_w).grid(row=1, column=1)

self.tab1\_entry\_y2 = ttk.Entry(self.tab1, textvariable=self.var4\_1, width=self.e\_w).grid(row=1, column=3)

self.tab1.grid\_rowconfigure(0, weight=1)

self.tab1.grid\_rowconfigure(1, weight=1)

self.tab1.grid\_columnconfigure(0, weight=1)

self.tab1.grid\_columnconfigure(1, weight=1)

# Tab 2

self.tab2\_label\_1 = ttk.Label(self.tab2, text="X:").grid(row=0, column=0)

self.tab2\_label\_2 = ttk.Label(self.tab2, text=" Y:").grid(row=0, column=2)

self.tab2\_label\_3 = ttk.Label(self.tab2, text="L").grid(row=1, column=0)

self.tab2\_label\_4 = ttk.Label(self.tab2, text="Angle:").grid(row=1, column=2)

self.tab2\_entry\_x = ttk.Entry(self.tab2, textvariable=self.var1\_2, width=self.e\_w).grid(row=0, column=1)

self.tab2\_entry\_y = ttk.Entry(self.tab2, textvariable=self.var2\_2, width=self.e\_w).grid(row=0, column=3)

self.tab2\_entry\_a = ttk.Entry(self.tab2, textvariable=self.var3\_2, width=self.e\_w).grid(row=1, column=1)

self.tab2\_entry\_b = ttk.Entry(self.tab2, textvariable=self.var4\_2, width=self.e\_w).grid(row=1, column=3)

self.tab2.grid\_rowconfigure(0, weight=1)

self.tab2.grid\_rowconfigure(1, weight=1)

self.tab2.grid\_columnconfigure(0, weight=1)

self.tab2.grid\_columnconfigure(1, weight=1)

self.button\_line\_color = ttk.Button(self.work, text="Цвет линии",

command=lambda: get\_color\_line(self)).grid(row=1, column=0)

self.label\_line\_color = tk.Label(self.work, bg=self.color\_line, width=20)

self.label\_line\_color.grid(row=1, column=1)

self.button\_bg\_color = ttk.Button(self.work, text="Цвет фона",

command=lambda: get\_color\_bg(self)).grid(row=2, column=0)

self.label\_bg\_color = tk.Label(self.work, bg=self.color\_bg, width=20)

self.label\_bg\_color.grid(row=2, column=1)

self.ch\_var = tk.IntVar()

self.ch\_button\_color = ttk.Checkbutton(self.work, text="Рисовать цветом фона", variable=self.ch\_var)

self.ch\_var.set(0)

self.ch\_button\_color.grid(row=3, column=0)

self.list\_alg = ["ЦДА", "Брезенхема(Целые)", "Брезенхема(Действительные)",

"Брезенхема(Без ступенчатности)", "Алгоритм ВУ", "Библиотичный метод"]

self.combobox\_alg = ttk.Combobox(self.work, width=25, values=self.list\_alg, state="readonly")

self.combobox\_alg.current(0)

self.combobox\_alg.grid(row=4, column=0)

self.button\_test = ttk.Button(self.work, text="Времия",

command=lambda: time\_test(100000)).grid(row=5, column=0)

self.button\_stepping = ttk.Button(self.work, text = 'Ступенчатность',

command = lambda:stepping()).grid(row = 5, column = 1)

self.button\_draw = ttk.Button(self.work, text="Нарисовать",

command=lambda: draw(self)).grid(row=6, column=0)

self.button\_clear = ttk.Button(self.work, text="Очистить",

command=lambda: clear(self) ).grid(row = 6, column=1)

self.work.pack(side=tk.RIGHT)

self.canvas = tk.Canvas(self, bg=self.color\_bg)

self.canvas.pack(side=tk.LEFT, fill=tk.BOTH, expand = True)

# Graph page class

class GraphPage(tk.Frame):

def \_\_init\_\_(self, parent, controller):

tk.Frame.\_\_init\_\_(self, parent)

button1 = ttk.Button(self, text="Вернуться",

command=lambda: controller.show\_frame(MainPage))

button1.pack()

def clear(self):

self.canvas.delete("all")

self.canvas.configure(bg=self.color\_bg)

def mes(text):

messagebox.showinfo("Внимание", text)

def get\_color\_line(self):

color = askcolor()[1]

if color:

self.color\_line = color

self.label\_line\_color.configure(bg = color)

def get\_color\_bg(self):

color = askcolor()[1]

if color:

self.color\_bg = color

self.label\_bg\_color.configure(bg = color)

self.canvas.configure(bg = self.color\_bg)

def draw(self):

tab = self.tab\_parent.index(self.tab\_parent.select())

if tab == 0:

try:

x1 = int(self.var1\_1.get())

y1 = int(self.var2\_1.get())

x2 = int(self.var3\_1.get())

y2 = int(self.var4\_1.get())

except ValueError:

mes("Неверные данные")

return -1

if tab == 1:

try:

x = int(self.var1\_2.get())

y = int(self.var2\_2.get())

r = int(self.var3\_2.get())

angle = int(self.var4\_2.get())

except ValueError:

mes("Неверные данные")

return -1

if int(self.ch\_var.get()) == 1:

color = self.color\_bg

else:

color = self.color\_line

alg = self.combobox\_alg.current()

can\_x = self.canvas.winfo\_width()

can\_y = self.canvas.winfo\_height()

if tab == 0:

line = Line(x1, y1, x2, y2, color, alg)

draw\_func(self, line)

if tab == 1:

spectrum = Spectrum(x, y, r, angle, color, alg)

draw\_func(self, spectrum)

def draw\_spectrum(self, x0, y0, r, angle, alg, color):

grad = get\_rgb\_intensity(self, self.color\_line, self.color\_bg, INTENSITY)

step = int(360 / angle)

for i in range(step):

cur = (angle \* pi / 180) \* i

# cur += ang

x = int(round(x0 + r \* cos(cur)))

y = int(round(y0 + r \* sin(cur)))

if alg == 0:

drawline\_dda(self, x0, y0, x, y)

elif alg == 1:

drawline\_br\_float(self, x0, y0, x, y)

elif alg == 2:

drawline\_br\_int(self, x0, y0, x, y)

elif alg == 3:

drawline\_br\_smooth(self, x0, y0, x, y, grad, INTENSITY)

elif alg == 4:

drawline\_wu(self, x0, y0, x, y, grad, INTENSITY)

elif alg == 5:

drawline\_lib(self, x0, y0, x, y)

INTENSITY = 100

def draw\_func(self, obj):

self.color\_pen = obj.color

grad = get\_rgb\_intensity(self, self.color\_line, self.color\_bg, INTENSITY)

if type(obj) == Line:

if obj.alg == 0:

drawline\_dda(self, obj.x1, obj.y1, obj.x2, obj.y2)

elif obj.alg == 1:

drawline\_br\_float(self, obj.x1, obj.y1, obj.x2, obj.y2)

elif obj.alg == 2:

drawline\_br\_int(self, obj.x1, obj.y1, obj.x2, obj.y2)

elif obj.alg == 3:

drawline\_br\_smooth(self, obj.x1, obj.y1, obj.x2, obj.y2, grad, INTENSITY)

elif obj.alg == 4:

drawline\_wu(self, obj.x1, obj.y1, obj.x2, obj.y2, grad, INTENSITY)

elif obj.alg == 5:

drawline\_lib(self, obj.x1, obj.y1, obj.x2, obj.y2)

elif type(obj) == Spectrum:

draw\_spectrum(self, obj.x, obj.y, obj.r, obj.angle, obj.alg, obj.color)

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

app = Kg4App(1000, 600)

app.mainloop()