**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**Національний технічний університет України**

**«Київський Політехнічний Інститут»**

*Факультет інформатики та обчислювальної техніки*

*Кафедра обчислювальної техніки*

**Лабораторна робота №2**

*з дисципліни «Комп’ютерна графіка»*

*на тему: «Греометричні моделі»*

**Виконали:**

студенти 2-го курсу ФІОТ

групи ІВ-82

*Данилюк Д. А.*

*Борозенець Д. Р.*

Бригада: №5

**Перевірив:**

Старший викладач

*Саверченко В. Г.*

**Київ – 2019 р.**

**ЛАБОРАТОРНА РОБОТА №2**

*Графічні примітиви*

**Мета:** Навчитися будувати складні геометричні орнаменти з простих графічних примітивів

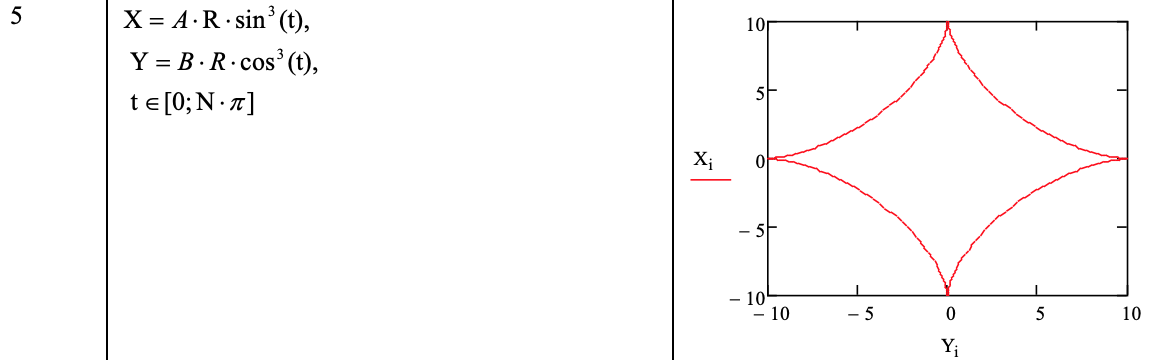
1. **Завдання**

1. Построить геометрическую модель M1 согласно варианту заданий. Для этого определить конкретные значения параметров (R, , N, A, B, D, m, K, H) и построить соответствующую ей графическую модель.

2. Разработать модель орнамента M2(М1,P2, … , PN, где М1- геометрическая модель по варианту, а P2, … , PN – параметры модели, определяющие конфигурацию, количество, размер, шаг тиражирования графической модели и др.

3. Разработать модель визуального спецэффекта M3(M2,С1, С2, … , PК), где M2 – модель орнамента, а Сj – параметры модели визуального спецэффекта (муар, пульсация и т.п.).

4. Выводы по работе должны содержать результаты исследований для трех разработанных моделей (описание моделей со значениями их параметров)

****

**ІІ. Код програми**

main.py

import argparse  
import math  
import tkinter as tk  
from lab\_2.figure import Figure as figure  
  
def main(n, m):  
 window = tk.Tk()  
 window.title("lab #2")  
 # VARIABLES  
 screenSize = (700, 700) # (width, height)  
 canvas = tk.Canvas(window, width=screenSize[0], height=screenSize[1])  
 # Here is 4 pre-sets (choose from 1 to 5) or do your own figure  
 preSetNumber = 5  
  
 if preSetNumber == 1:  
 mdl = figure(20, 10, 15, 5, (350, 350))  
 mdl.createFigure(canvas, 0, "Red")  
  
 elif preSetNumber == 2:  
 n = 5  
 m = 5  
 mdl1 = figure(20, 20, 6, 5, (350, 350))  
 mdl2 = figure(20, 20, 12, 5, (350, 350))  
 for angle in [math.pi / n \* i for i in range(-n, n)]:  
 mdl1.createFigure(canvas, angle, "Blue", turnPoint=0)  
  
 for angle in [math.pi / n \* i for i in range(-m, m)]:  
 mdl2.createFigure(canvas, angle, "Orange", turnPoint=0)  
  
 elif preSetNumber == 3:  
 n = 4  
 mdl = figure(20, 20, 6, 5, (230, 230))  
 for angle in [math.pi / n \* i for i in range(-n, n)]:  
 mdl.createFigure(canvas, angle, "Navy Blue", turnPoint=1)  
  
 elif preSetNumber == 4:  
 n = 3  
 mdl = figure(20, 20, 6, 5, (250, 250))  
 for angle in [math.pi / n \* i for i in range(-n, n)]:  
 mdl.createFigure(canvas, angle, "Blue", turnPoint=2)  
  
 elif preSetNumber == 5:  
 n = 60  
 mdl = figure(20, 20, 9, 5, (250, 250))  
 for angle in [math.pi / n \* i for i in range(-n, n)]:  
 mdl.createFigure(canvas, angle, "magenta", turnPoint=2)  
  
 window.mainloop()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 parser = argparse.ArgumentParser(  
 'Built different charts with figures according to params')  
 parser.add\_argument(  
 '--n', help='Initialize num for first figure rotation', type=int, default=4)  
 parser.add\_argument(  
 '--m', help='Initialize num for second figure rotation', type=int, default=5)  
  
 args = parser.parse\_args()  
 main(args.n, args.m)

figure.py

import math  
import numpy  
  
class Figure:  
  
 def \_\_init\_\_(self, sizeA: int, sizeB: int, sizeR: int, sizeN: int, centerPoint: tuple):  
 self.sizeA = sizeA  
 self.sizeB = sizeB  
 self.sizeR = sizeR  
 self.sizeN = sizeN  
 self.centerPoint = centerPoint  
  
 def getXY(self) -> list:  
 allPoints: list = []  
 for t in numpy.arange(0.0, self.sizeN \* math.pi, 0.1):  
 x = self.sizeA \* self.sizeR \* pow(math.sin(t), 3)  
 y = self.sizeB \* self.sizeR \* pow(math.cos(t), 3)  
 x += self.centerPoint[0]  
 y += self.centerPoint[1]  
  
 allPoints.append((x, y))  
  
 return allPoints  
  
 def turning\_point\_1(self) -> tuple:  
 x = self.centerPoint[0] + self.sizeA \* self.sizeR  
 y = self.centerPoint[1] + self.sizeB \* self.sizeR  
 return x, y  
  
 def turning\_point\_2(self) -> tuple:  
 x = self.centerPoint[0] + (self.sizeA \* self.sizeR / 2)  
 y = self.centerPoint[1] + (self.sizeB \* self.sizeR / 2)  
 return x, y  
  
 def turning\_point\_center(self) -> tuple:  
 x = self.centerPoint[0]  
 y = self.centerPoint[1]  
 return x, y  
  
 def \_transform(self, x: tuple, y: tuple, turnPoint: tuple, angle: float) -> tuple:  
 x -= turnPoint[0]  
 y -= turnPoint[1]  
  
 temp\_x = x \* math.cos(angle) - y \* math.sin(angle)  
 temp\_y = x \* math.sin(angle) + y \* math.cos(angle)  
  
 return temp\_x + turnPoint[0], temp\_y + turnPoint[1]  
  
 def rotate(self, angle: float, turnPoint: int = 0) -> list:  
 turnedPoint = (self.turning\_point\_center(), self.turning\_point\_1(), self.turning\_point\_2())[turnPoint]  
  
 rotated\_coordinates = [  
 self.\_transform(x, y, turnedPoint, angle) for x, y in self.getXY()  
 ]  
  
 return rotated\_coordinates  
  
 def createFigure(self, canvas, angle: float, color: str = "Black", turnPoint: int = 0):  
 m = self.rotate(angle, turnPoint=turnPoint)  
  
 for i in range(len(m) - 1):  
 canvas.create\_line(m[i][0], m[i][1], m[i + 1][0], m[i + 1][1], fill=color)  
  
 canvas.pack()

**ІІІ. Результат**

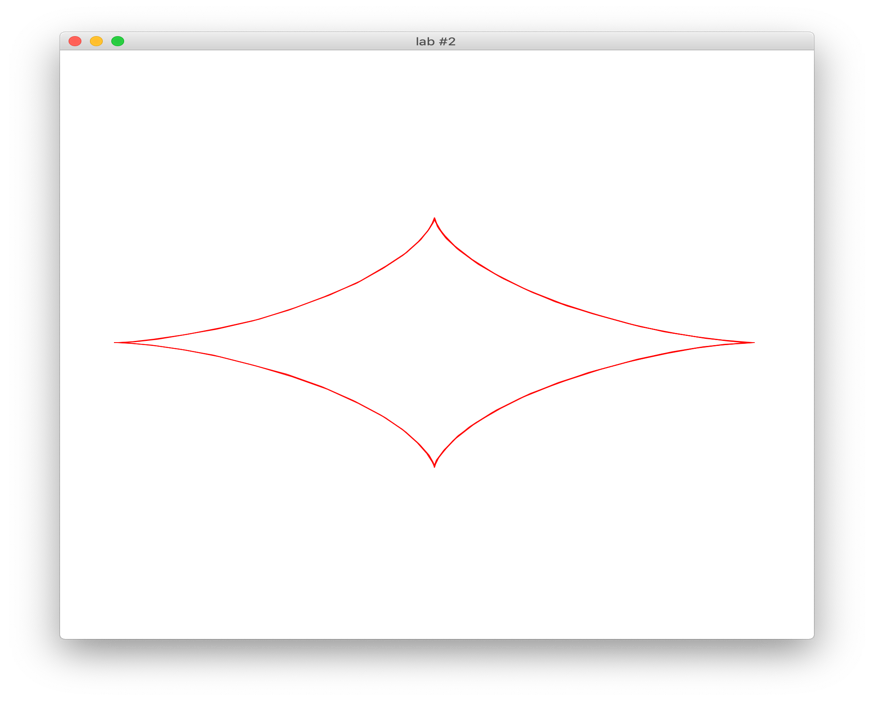
model = figure(A, B, R, N, (centerPoint))

centerPoint = (x, y)

n = кількість маленьких фігур

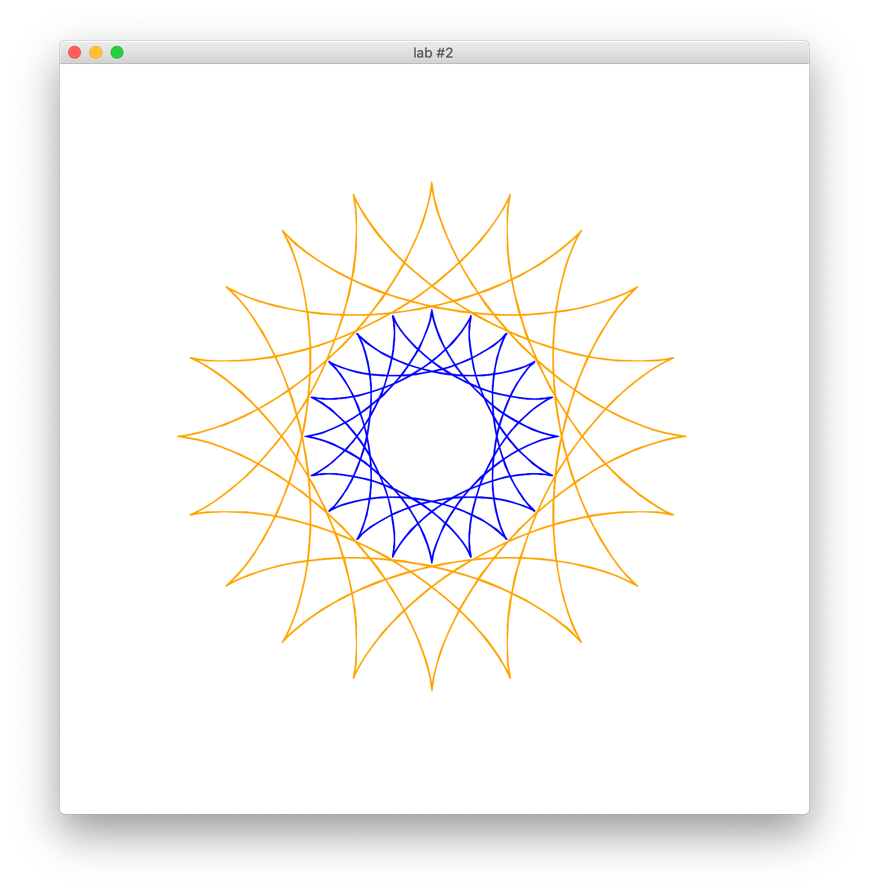
m = кількість великих фігур

№ 1 (Фігура)



mdl = figure(20, 10, 15, 5, (350, 350))  
mdl.createFigure(canvas, 0, "Red")

№ 2 (Орнамент №1)

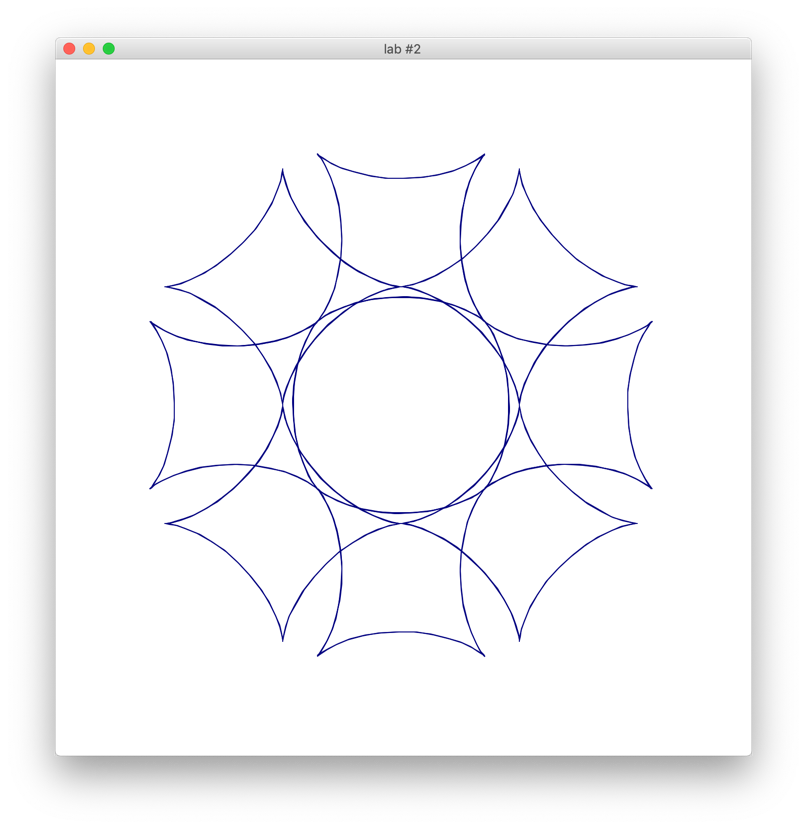


n = 5  
m = 5

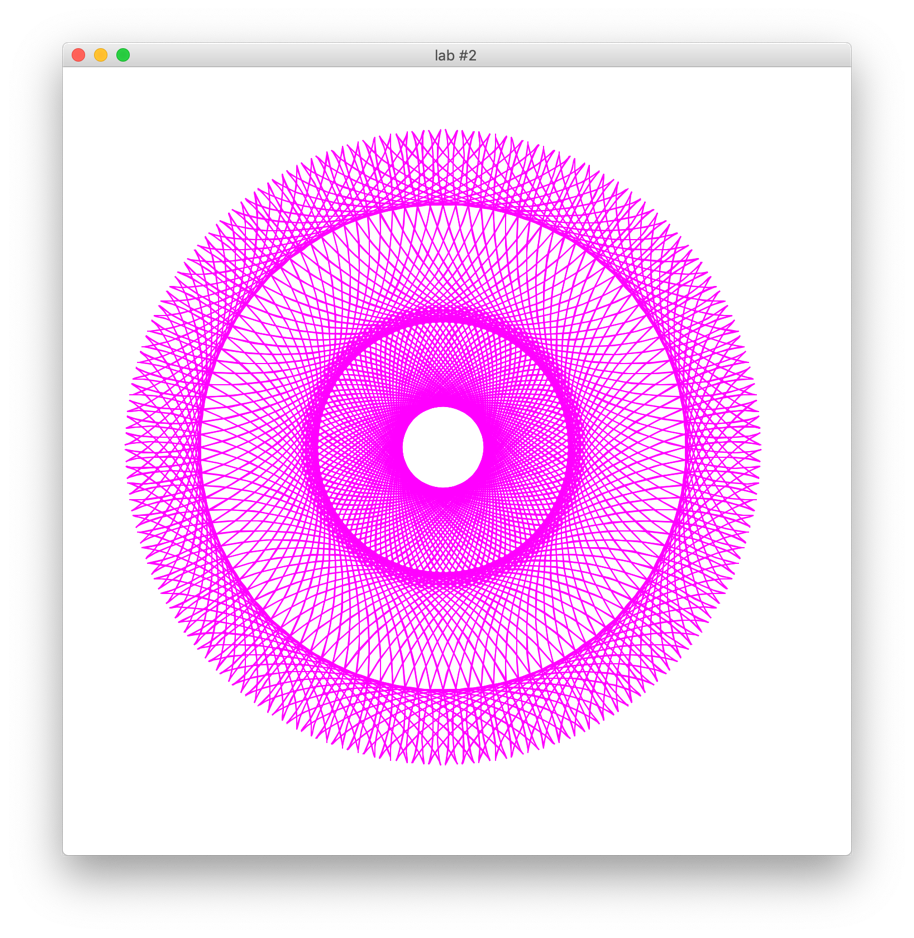
mdl1 = figure(20, 20, 6, 5, (350, 350))  
mdl2 = figure(20, 20, 12, 5, (350, 350))  
  
for angle in [math.pi / n \* i for i in range(-n, n)]:  
 mdl1.createFigure(canvas, angle, "Blue", turnPoint=0)

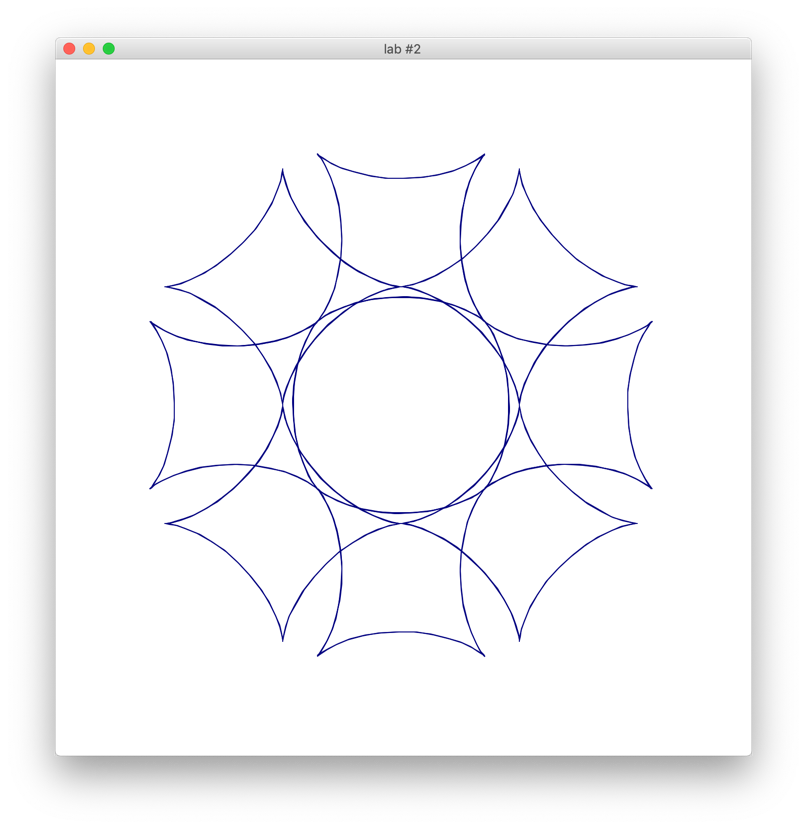
for angle in [math.pi / n \* i for i in range(-m, m)]:  
 mdl2.createFigure(canvas, angle, "Orange", turnPoint=0)

№ 3 (Орнамент №2)



n = 4  
  
mdl = figure(20, 20, 6, 5, (230, 230))  
  
for angle in [math.pi / n \* i for i in range(-n, n)]:  
 mdl.createFigure(canvas, angle, "Navy Blue", turnPoint=1)

№ 4 (Муар)



n = 60  
  
mdl = figure(20, 20, 9, 5, (250, 250))  
  
for angle in [math.pi / n \* i for i in range(-n, n)]:  
 mdl.createFigure(canvas, angle, "magenta", turnPoint=2)

**ІV. Висновок**

У ході лабораторної роботи була створена програма, яка малює геометричну модель. Для малювання використано бібліотеку tkinter мови python.