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: «Геометричні 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

I. Завдання

1. Построить геометрическую модель М1 согласно варианту заданий. Для этого определить конкретные значения параметров (R, ω, N, A, B, D, m, K, H) и построить соответствующую ей графическую модель.

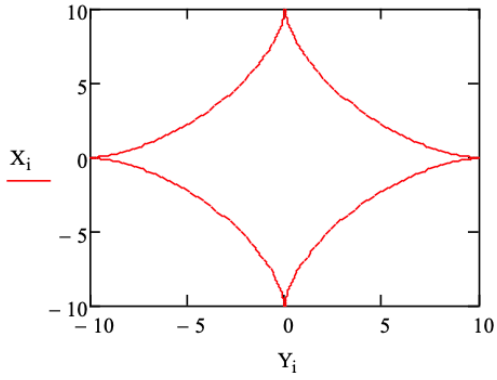
2. Разработать модель орнамента М2(М1,Р2, … , PN, где М1- геометрическая модель по варианту, а Р2, … , PN – параметры модели, определяющие конфигурацию, количество, размер, шаг тиражирования графической модели и др.

3. Разработать модель визуального спецэффекта М3(М2,C_1, C_2, … , P_K), где М2 – модель орнамента, а C_j – 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)

5

$$X = A \cdot R \cdot \sin^3(t),$$
$$Y = B \cdot R \cdot \cos^3(t),$$
$$t \in [0; N \cdot \pi]$$



II. 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

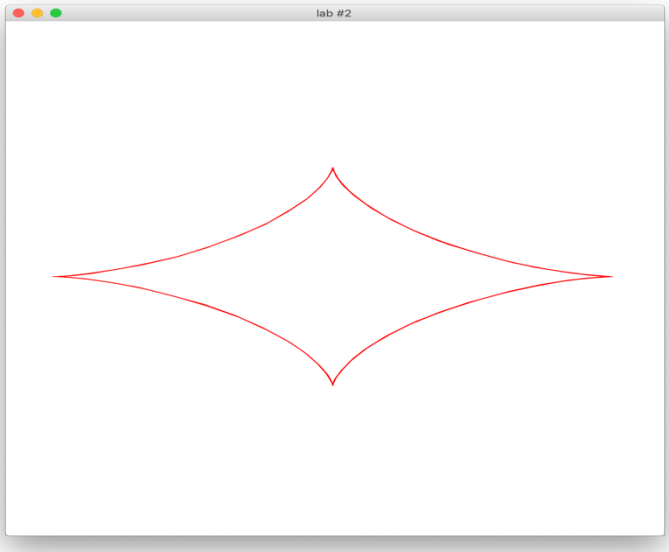
```python
import math
import numpy

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()
```

III. Результат

model = figure(A, B, R, N, (centerPoint))
centerPoint = (x, y)
n = кількість маленьких фігур
m = кількість великих фігур

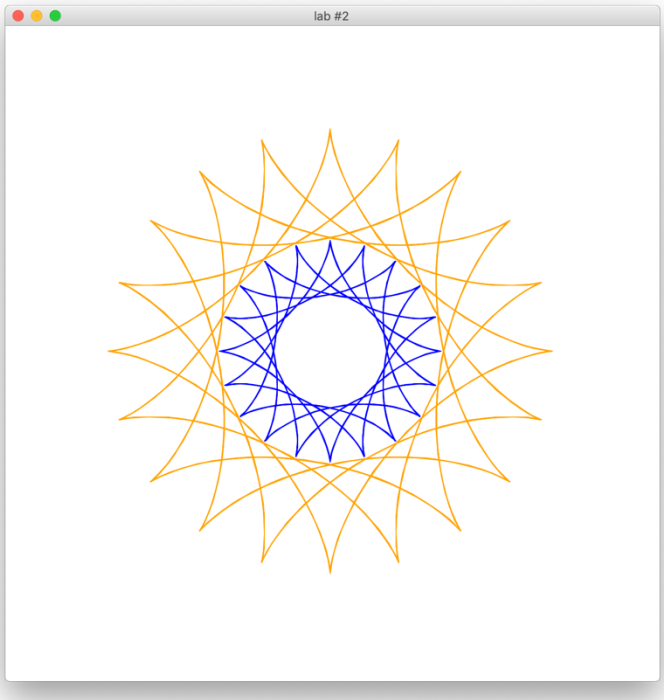
№ 1 (Фігура)

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



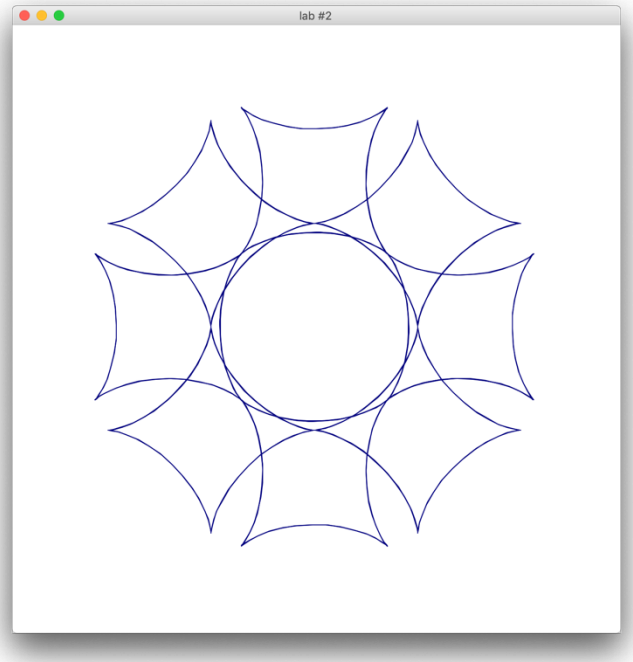
№ 2 (Орнамент №1)

```
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)
```



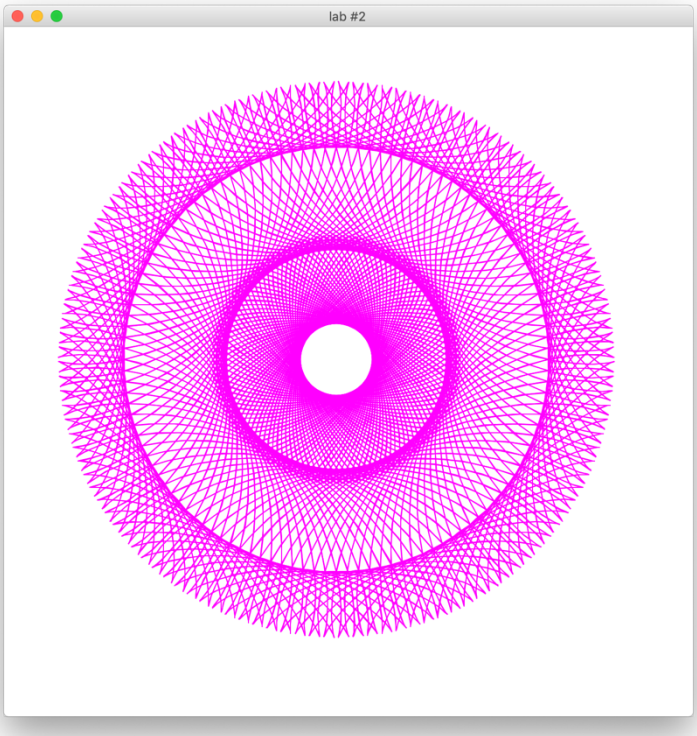
№ 3 (Орнамент №2)

```
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)
```



IV. Висновок

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