МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний технічний університет України
«Київський Політехнічний Інститут»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №1

з дисципліни «Комп'ютерна графіка»

на тему: «Графічні примітиви»

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 №1

Графічні примітиви

Мета: Навчитися будувати складні геометричні орнаменти з простих графічних примітивів

I. Завдання

1. Построить модель базового элемента M1(A_1,A_2, … , A_N) на основе графических примитивов согласно варианту заданий, где A_i – параметры модели.

2. Разработать модель орнамента M2(M1,B_1, B_2, … , P_M), где M1 – модель базового элемента, а B_i – параметры модели орнамента, определяющие конфигурацию, количество, размер, шаг тиражирования и т.п. для базовых элементов.

3. Разработать модель визуального спецэффекта M3(M2,C_1, C_2, … , P_K), где M2 – 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

II. Код програми

main.py

```
import tkinter as tk
import math
import argparse
from lab_1 import SameSizeTriangle as model


def main(n, m):
    window = tk.Tk()

    # VARIABLES
    screenSize = (700, 700)  # (width, height)

    size1 = 400  # Mark: - Size of larger figure
    startPoint1 = (140, 120)  # (x, y)  Start point of larger figure
    color1 = "Orange"

    size2 = 110  # Mark: - Size of smaller figure
    color2 = "Navy Blue"

    '''
        Mark: - the start point coordinates of the smaller figure
        are calculated automatically (should be in the center)
        depending on the coordinates larger figure
    '''
    canvas = tk.Canvas(window, width=screenSize[0], height=screenSize[1])

    coordinates1 = [
        size1, (startPoint1[0], startPoint1[1])
    ]
    mdl1 = model(*coordinates1)

    coordinates2 = [
        size2, (mdl1.get_center_coord()[0] - (size2 / 2),
mdl1.get_center_coord()[1])
    ]
    mdl2 = model(*coordinates2)

    for angle in [math.pi / n * i for i in range(-n, n)]:
        mdl1.create_figure(canvas, angle, color1, center=True)

    for angle in [math.pi / m * i for i in range(-m, m)]:
        mdl2.create_figure(canvas, angle, color2, center=False)

    window.mainloop()


if __name__ == '__main__':
    parser = argparse.ArgumentParser(
        'Built different charts with sqaures acording to params')
    parser.add_argument(
        '--n', help='Initialize num for center rotation', type=int,
default=5)
    parser.add_argument(
        '--m', help='Initialize num for corner rotation', type=int,
default=12)

    args = parser.parse_args()
    main(args.n, args.m)
```

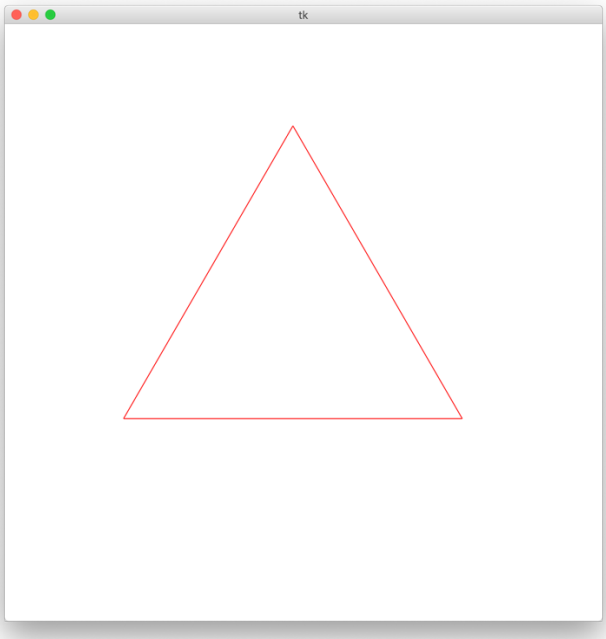
samesizetriangle.py

```python
import math


class SameSizeTriangle:
    """
    Default class for using in ornament
    """

    def __init__(self, size: int, startPoint: tuple):
        self.size = size
        self.startPoint = startPoint

    def get_coords(self) -> tuple:
        a1 = self.startPoint[0] + (self.size / 2), self.startPoint[1]
        a2 = self.startPoint[0] + self.size, \
             self.startPoint[1] + ((self.size * math.sqrt(3)) / 2)
        a3 = self.startPoint[0], self.startPoint[1] + ((self.size *
math.sqrt(3)) / 2)
        return a1, a2, a3

    def get_center_coord(self) -> tuple:
        x = self.get_coords()[0][0]
        y = self.get_coords()[0][1] + (self.size / math.sqrt(3))
        return x, y

    def _transform(self, x: tuple, y: tuple, center: tuple, angle: float) ->
tuple:
        x -= center[0]
        y -= center[1]

        temp_x = x * math.cos(angle) - y * math.sin(angle)
        temp_y = x * math.sin(angle) + y * math.cos(angle)

        return temp_x + center[0], temp_y + center[1]

    def rotate(self, angle: float, center=True) -> list:
        center = (self.get_coords()[0], self.get_center_coord())[center]

        rotated_coordinates = [
            self._transform(x, y, center, angle) for x, y in
self.get_coords()
        ]

        return rotated_coordinates

    def creatr_figure(self, canv, angle: float, color: str, center: bool =
True, k: int = 0):
        m = self.rotate(angle, center=center)

        canv.create_line(m[0][0] - k, m[0][1] - k, m[1][0] - k, m[1][1] - k,
fill=color)
        canv.create_line(m[0][0] - k, m[0][1] - k, m[2][0] - k, m[2][1] - k,
fill=color)
        canv.create_line(m[1][0] - k, m[1][1] - k, m[-1][0] - k, m[-1][1] -
k, fill=color)

        canv.pack()
```

III. Результат

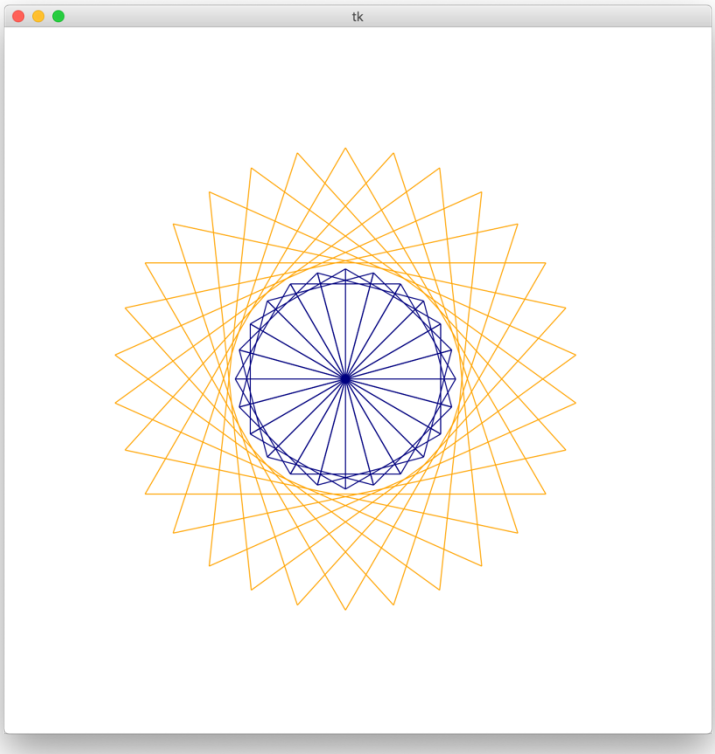
1. Модель базового елемента

M1(size, startPoint), d-довжина сторони, startPoint – точка початку
M1(400, (140, 120)):



2. Орнамент

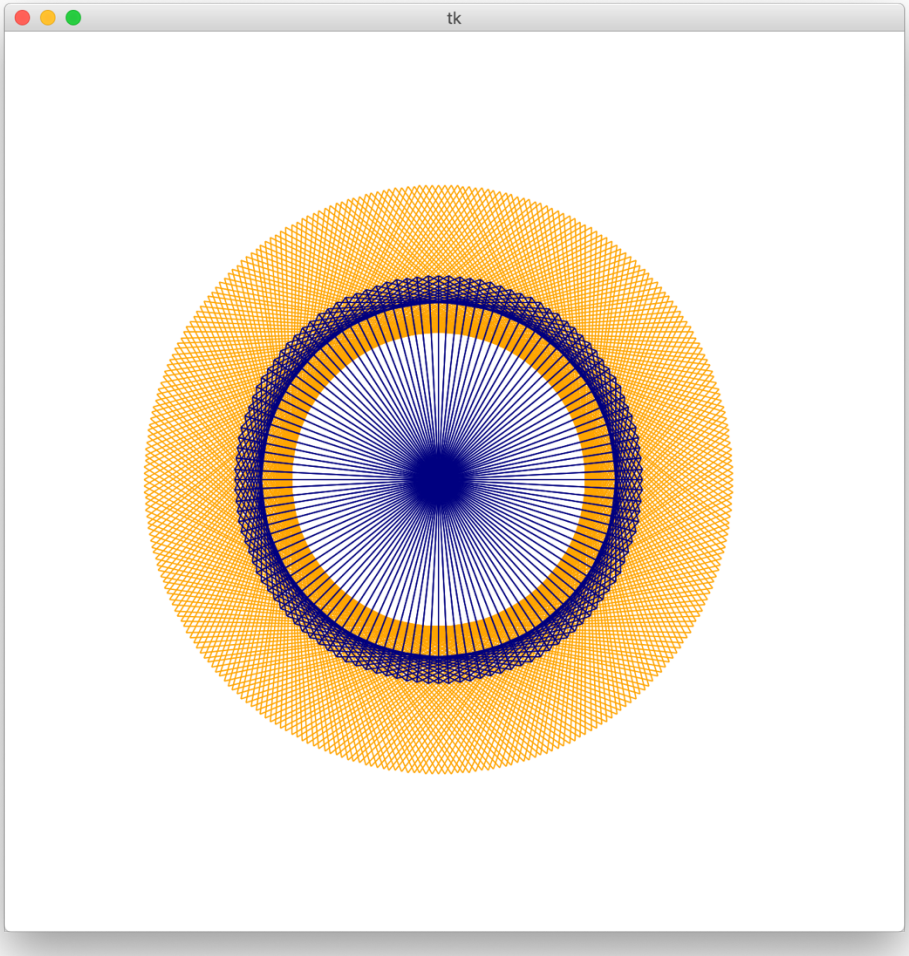
M2(M1(size, startPoint), n, m), m і n - кількість базових елементів.
M2(M1(400, (140, 120)), M1(110)) 5, 12):



3. Муар

M3(M1(d), m, n) m-кількість базових елементів.

M2(M1(400, (140, 120)), M1(110)), 50, 60):



IV. Висновок

У ході лабораторної роботи була створена програма, яка малює елемент «Трикутник». Для малювання використано бібліотеку tkinter мови python.