**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

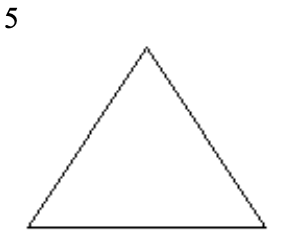
1. **Завдання**

1. Построить модель базового элемента M1(A1,A2, … , AN) на основе графических примитивов согласно варианту заданий, где Ai – параметры модели.

2. Разработать модель орнамента M2(M1,B1, B2, … , PM), где M1 – модель базового элемента, а Bi – параметры модели орнамента, определяющие конфигурацию, количество, размер, шаг тиражирования и т.п. для базовых элементов.

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 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

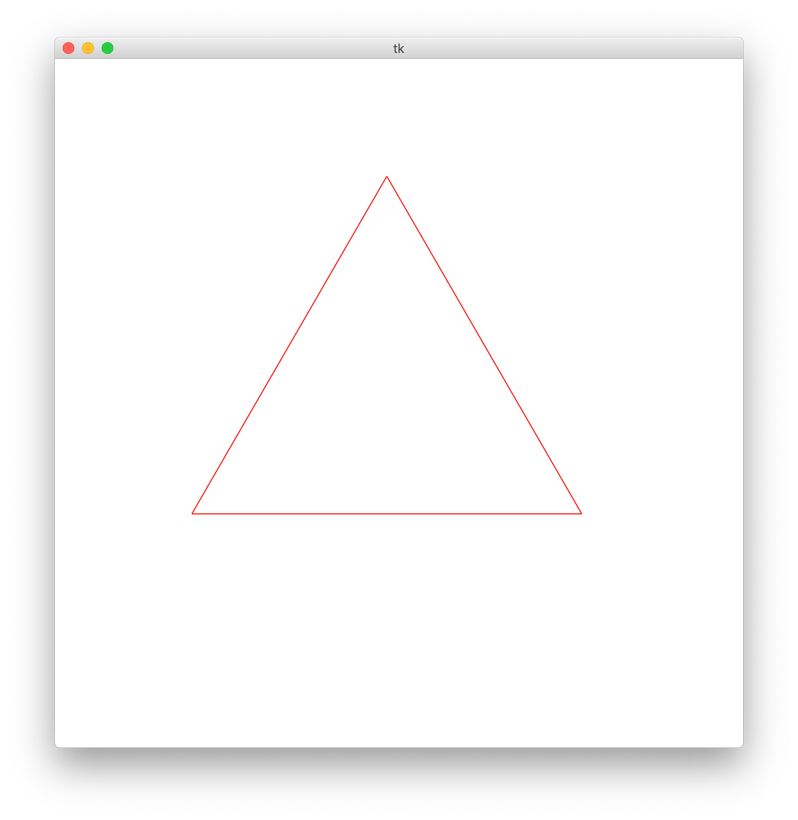
import math  
  
  
class SameSizeTriangle:  
 *"""  
 Default class for using in ornament  
 """* def \_\_init\_\_(self, size: int, startPoint: tuple):  
 self.size = size  
 self.startPoint = startPoint  
  
 def get\_coords(self) -> tuple:  
 a1 = self.startPoint[0] + (self.size / 2), self.startPoint[1]  
 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()

**ІІІ. Результат**

1. Модель базового елемента

M1(size, startPoint), d-довжина сторони, startPoint – точка початку

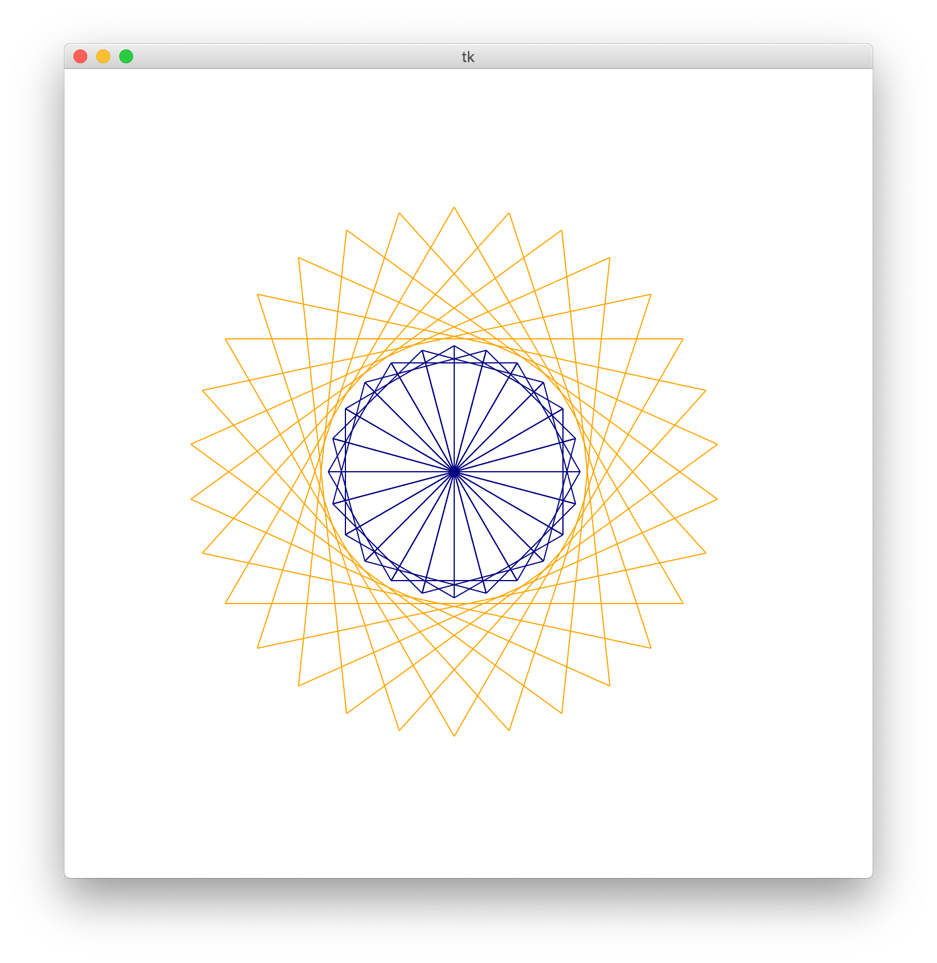
M1(400, (140, 120)):



1. Орнамент

M2(M1(size, startPoint), n, m), m і n - кількість базових елементів.

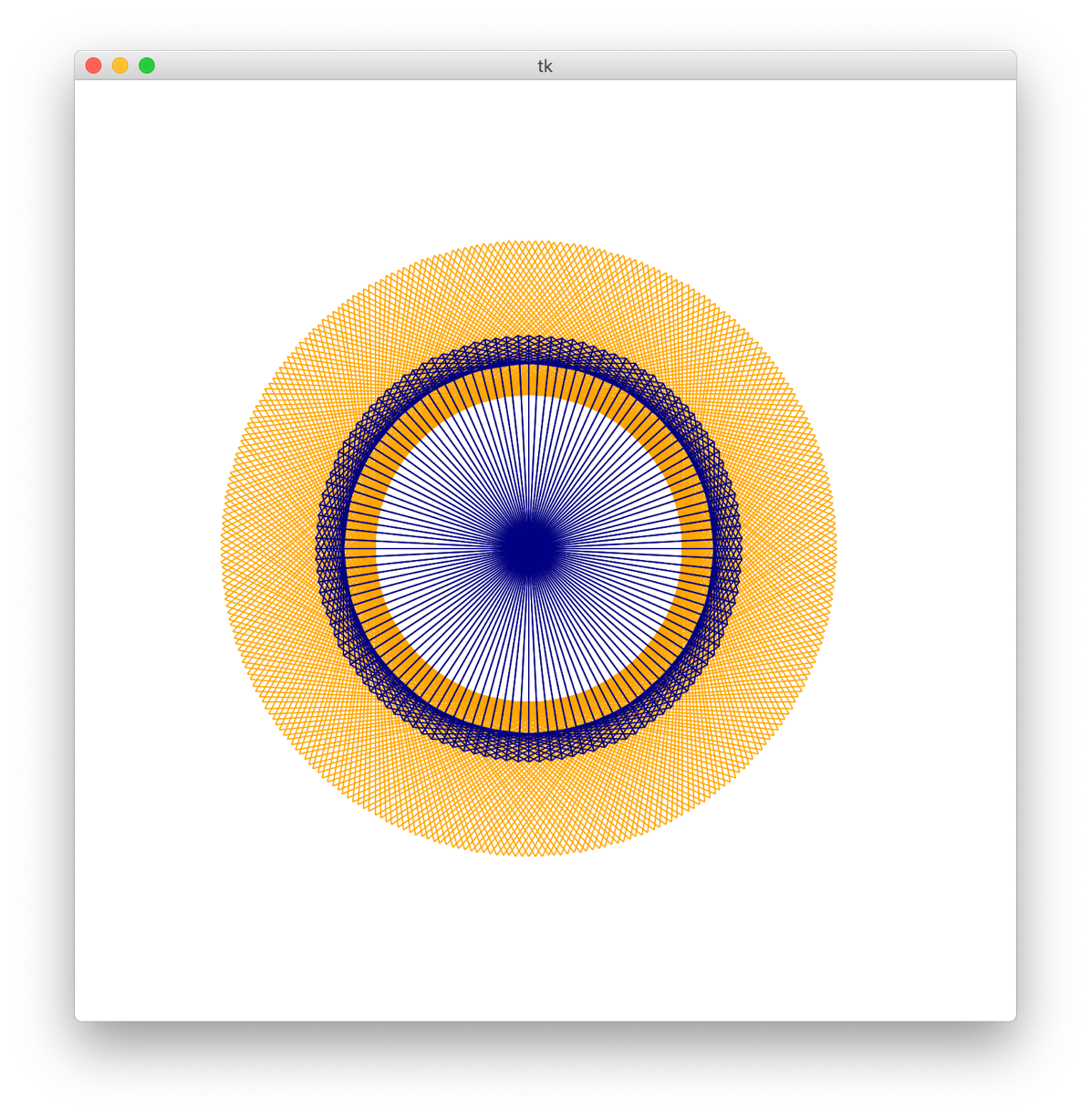
М2(M1(400, (140, 120)), M1(110)) 5, 12):



1. Муар

M3(М1(d), m, n) m-кількість базових елементів.

М2(M1(400, (140, 120)), M1(110)), 50, 60):



**ІV. Висновок**

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