**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

*Гевеленко Н. Р.*

*Захарчук Д. С.*

Бригада: №2

**Перевірив:**

Старший викладач

*Саверченко В. Г.*

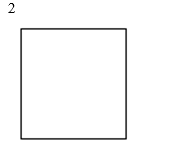
**Київ – 2019 р.**

**ЛАБОРАТОРНА РОБОТА №1**

*Графічні примітиви*

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

**I. Завдання**

****

**ІІ. Код програми**

chart.py

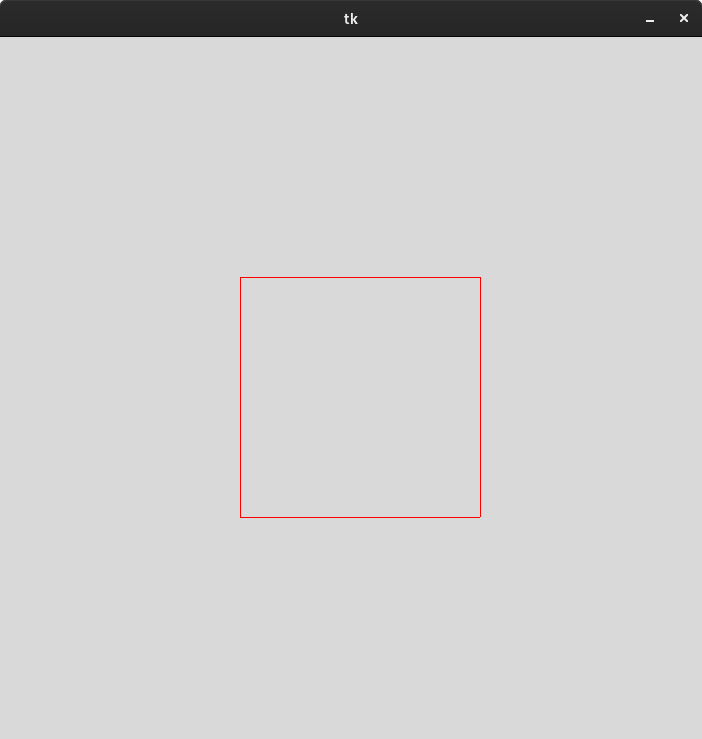
|  |
| --- |
|  |
|  |
|  | import math |
|  | import argparse |
|  | from square import Square as model |
|  | import tkinter as tk |
|  |  |
|  | def create\_square( |
|  | canv: tk.Canvas, mdl: model, |
|  | angle: float, color: str, |
|  | center: bool=True, k: int=0 |
|  | ): |
|  | m = mdl.rotate(angle, 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[2][0]-k, m[2][1]-k, m[-1][0]-k, m[-1][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() |
|  |  |
|  |  |
|  | def main(n, m, colors\_n, colors\_m): |
|  | window = tk.Tk() |
|  |  |
|  | SCALE = 1.2 |
|  | coords = [ # (x1, y1), (x2, y2), (x3, y3), (x4, y4) |
|  | (100 \* SCALE, 100 \* SCALE), (100 \* SCALE, 300 \* SCALE), |
|  | (300 \* SCALE, 100 \* SCALE), (300 \* SCALE, 300 \* SCALE) |
|  | ] |
|  |  |
|  | mdl = model(\*coords) |
|  | canv = tk.Canvas(window, width=700, height=700) |
|  | switch = True |
|  |  |
|  | for angle in [math.pi / m \* i for i in range(-m, m)]: |
|  | create\_square( |
|  | canv, mdl, angle, colors\_m[switch], center=False) |
|  | switch = not switch |
|  |  |
|  | for angle in [math.pi / n \* i for i in range(-n, n)]: |
|  | create\_square( |
|  | canv, mdl, angle, colors\_n[switch], k=-120) |
|  | switch = not switch |
|  |  |
|  | 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=10) |
|  | parser.add\_argument( |
|  | '--m', help='Initialize num for corner rotation', type=int, default=10) |
|  | parser.add\_argument( |
|  | '--colors\_inner', help='Initialize colors for center rotation', |
|  | type=str, default=['black', 'black'], nargs=2) |
|  | parser.add\_argument( |
|  | '--colors\_outer', help='Initialize colors for corner rotation', |
|  | type=str, default=['black', 'black'], nargs=2) |
|  |  |
|  | args = parser.parse\_args() |
|  | main(args.n, args.m, args.colors\_inner, args.colors\_outer) |
|  | square.py   |  | | --- | |  | |  |  | |  |  | |  | class Square(): | |  | ''' | |  | Default class for using in ornament | |  | ''' | |  | def \_\_init\_\_( | |  | self, a1: tuple, a2: tuple, a3: tuple, a4: tuple | |  | ): | |  | self.a1 = a1 | |  | self.a2 = a2 | |  | self.a3 = a3 | |  | self.a4 = a4 | |  |  | |  | def get\_coords(self) -> tuple: | |  | return self.a1, self.a2, self.a3, self.a4 | |  |  | |  | def \_get\_diag\_coords(self) -> tuple: | |  | return self.a1[0], self.a1[1], self.a4[0], self.a4[1] | |  |  | |  | def \_get\_center\_coords(self) -> tuple: | |  | diag = self.\_get\_diag\_coords() | |  | return (diag[0] + diag[2]) / 2, (diag[1] + diag[3]) / 2 | |  |  | |  | def \_get\_corner\_coords(self) -> tuple: | |  | diag = self.\_get\_diag\_coords() | |  | return diag[-2], diag[-1] | |  |  | |  | 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\_corner\_coords(), self.\_get\_center\_coords())[center] | |  |  | |  | rotated\_coords = [ | |  | self.\_transform(x, y, center, angle) | |  | for x, y | |  | in self.get\_coords() | |  | ] | |  |  | |  | return rotated\_coords | |

**ІІІ. Результат**

1. Модель базового елемента

M1(d), d-довжина сторони

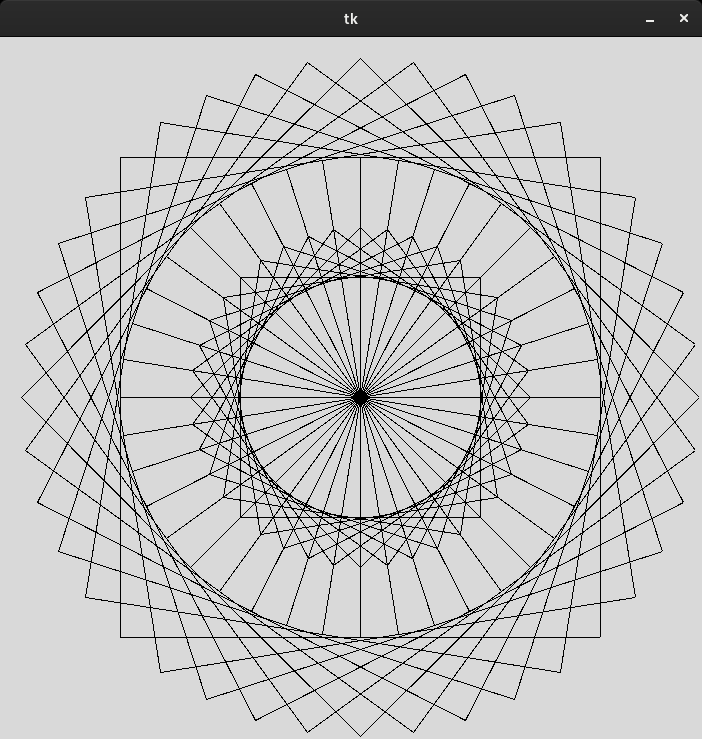
M1(200):



1. Орнамент

M2(М1(d), m, n), m і n- -кількість базових елементів.

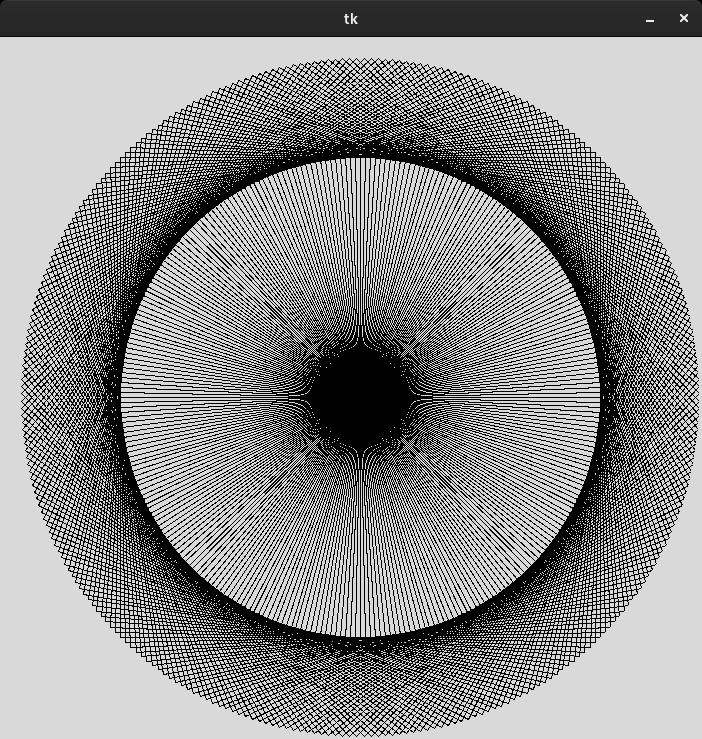
М2(М1(200), 20, 20):



1. Муар

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

M3(M1(200), 150, 0):



**ІV. Висновок**

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