МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Компьютерная графика»

ТЕМА: «ПОСТРОЕНИЕ ФРАКТАЛОВ»

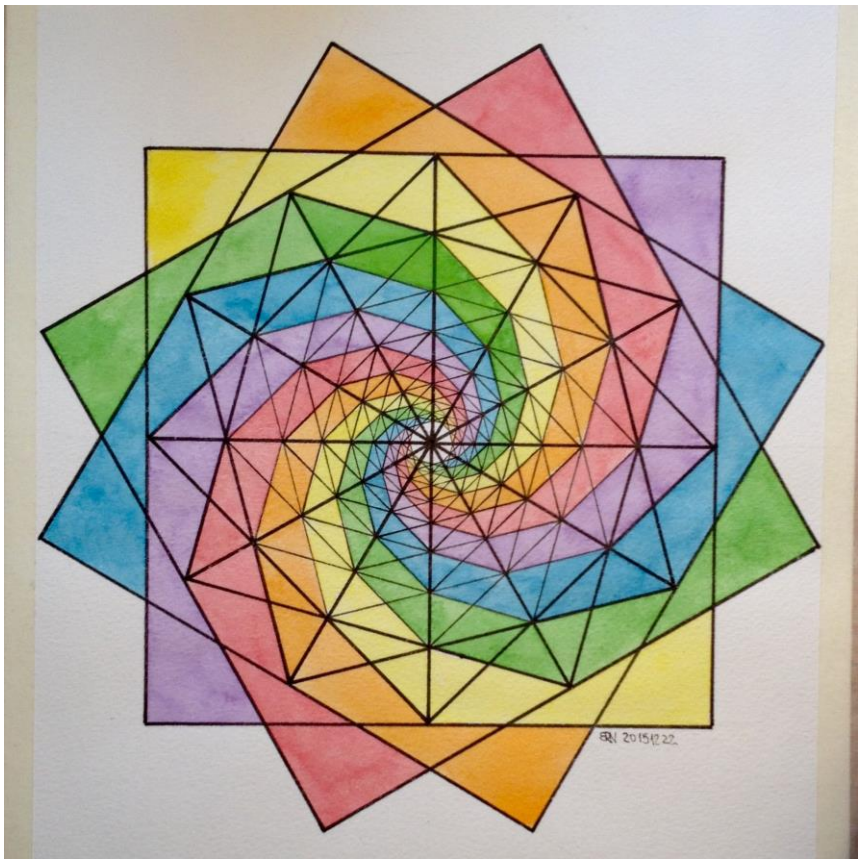
Усачева Д.В.
Студентки гр. 1384 _____ Пчелинцева К.Р.

Преподаватель _____ Герасимова Т.В.

Санкт-Петербург
2024

Задание.

Задание 1



Общие теоретические сведения.

Понятия фрактал и фрактальная геометрия, появившиеся в конце 70-х, с середины 80-х прочно вошли в обиход математиков и программистов. Слово фрактал образовано от латинского **fractus** и в переводе означает состоящий из фрагментов. Оно было предложено Бенуа Мандельбротом в 1975 году для обозначения нерегулярных, но самоподобных структур, которыми он занимался. Рождение фрактальной геометрии принято связывать с выходом в 1977 году книги Мандельброта `The Fractal Geometry of Nature'. В его работах использованы научные результаты других ученых, работавших в период 1875-1925 годов в той же области (Пуанкаре, Фату, Жюлиа, Кантор, Хаусдорф). Но только в наше время удалось объединить их работы в единую систему.

Фрактал (лат. fractus — дробленый) — термин, означающий геометрическую фигуру, обладающую свойством самоподобия, то есть

составленную из нескольких частей, каждая из которых подобна всей фигуре целиком.

Существует большое число математических объектов, называемых фракталами (треугольник Серпинского, снежинка Коха, кривая Пеано, множество Мандельброта). Фракталы с большой точностью описывают многие физические явления и образования реального мира: горы, облака, турбулентные (вихревые) течения, корни, ветви и листья деревьев, кровеносные сосуды, что далеко не соответствует простым геометрическим фигурам.

Выполнение работы.

Данная работа выполнена на операционной системе Windows 11. Приложение было создано на Python 3.11 с применением библиотеки OpenGL. Для создания интерфейса использовалась библиотека PyQt6. Интерфейс был разработан интерактивно в программе QtDesigner.

Окно приложение включает в себя виджет OpenGL и кнопки («-» и «+») с обработчиком нажатий. Так же есть информация о текущей фазе(шаге).

Фрактал, изображенный в задании, состоит из квадратов с диагоналями см. рисунок 1.

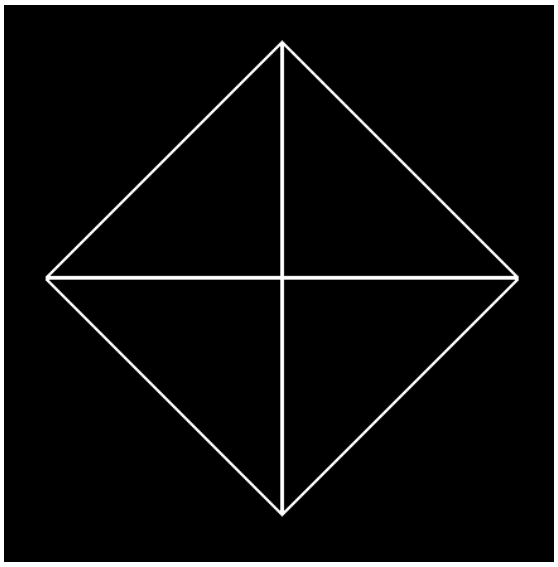


Рисунок 1 — Деталь фрактала

После тщательного рассмотрения фрактала-образца, было решено что три внешних квадрата не являются частью этого фрактала (тк не подобны остальным фрагментам). Для их отображения была написана отдельная функция. Также они были выделены красным цветом для наглядности см. рисунок 2.

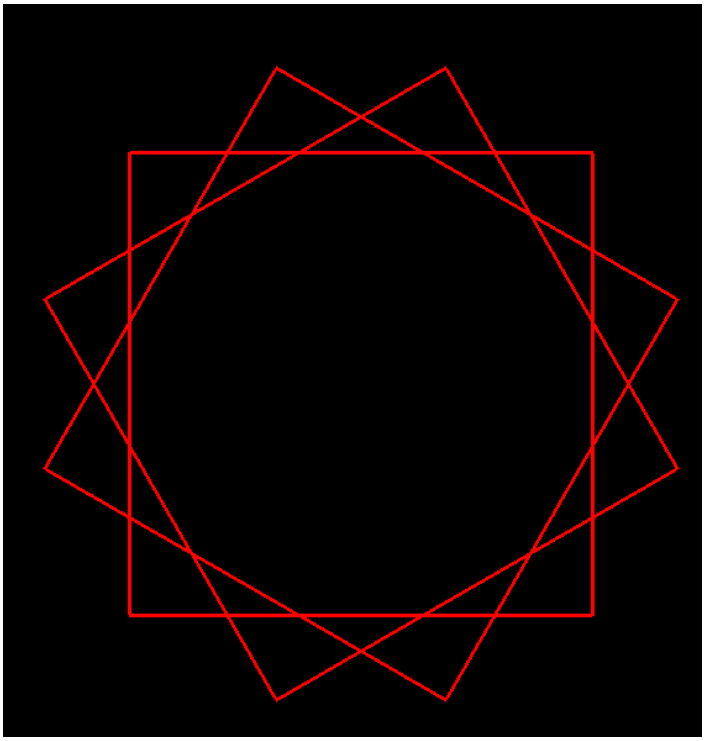


Рисунок 2 – Нулевая фаза

В дальнейших шагах число квадратов будет увеличиваться на 1. Следующая деталь получается из предыдущей при помощи поворота на 30 градусов. Также каждая третья деталь уменьшается в $\sqrt{2}$ (во столько раз квадрат вписанный в квадрат меньше) раз. Функции для создания новых деталей указаны ниже в листинге 1.

Листинг 1 — Главные действующие функции

```
        def fractal_step(self):
            for i in range(self.faze - 1):
                new_figure = np.array([self.rotate_dots(dot,
np.deg2rad(self.angle)) for dot in self.fractal[i][0]])
                if (i + 1) % 3 == 0:
                    self.fractal.append((new_figure *
self.coef,"GL_TRIANGLE_FAN"))
```

```
                else:
                    self.fractal.append((new_figure,"GL_TRIANGLE_FAN"))


        def not_a_fractal(self):
            new_figure = np.array([
                [ 0.5,  0.5],
                [-0.5,  0.5],
                [-0.5, -0.5],
                [ 0.5, -0.5]
            ])
            for i in range(3):
                new_figure = np.array([self.rotate_dots(dot,
np.deg2rad(self.angle)) for dot in new_figure])
                self.fractal.append((new_figure,"GL_QUADS"))
```

Тестирование.

Первая фаза представлена на рисунке 3.

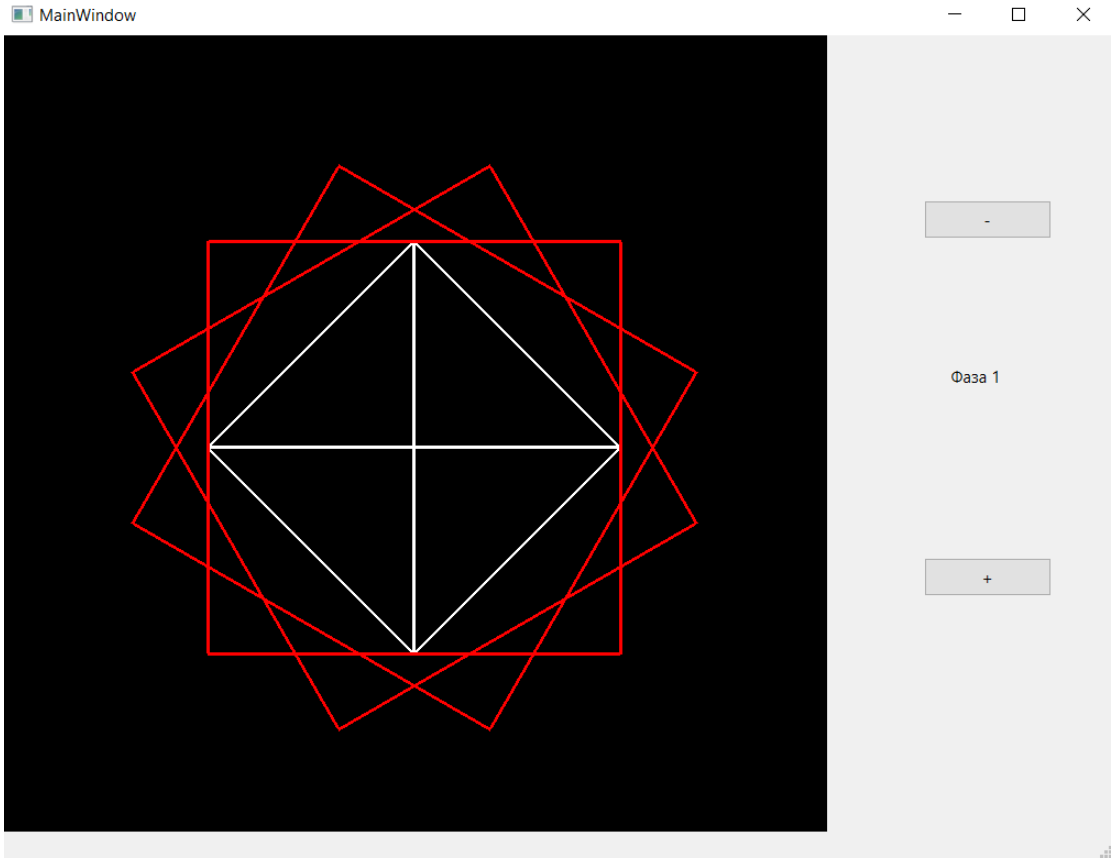


Рисунок 3 — 1 фаза

Наиболее похожее изображение достигается на 18 фазе см. рисунок 4.

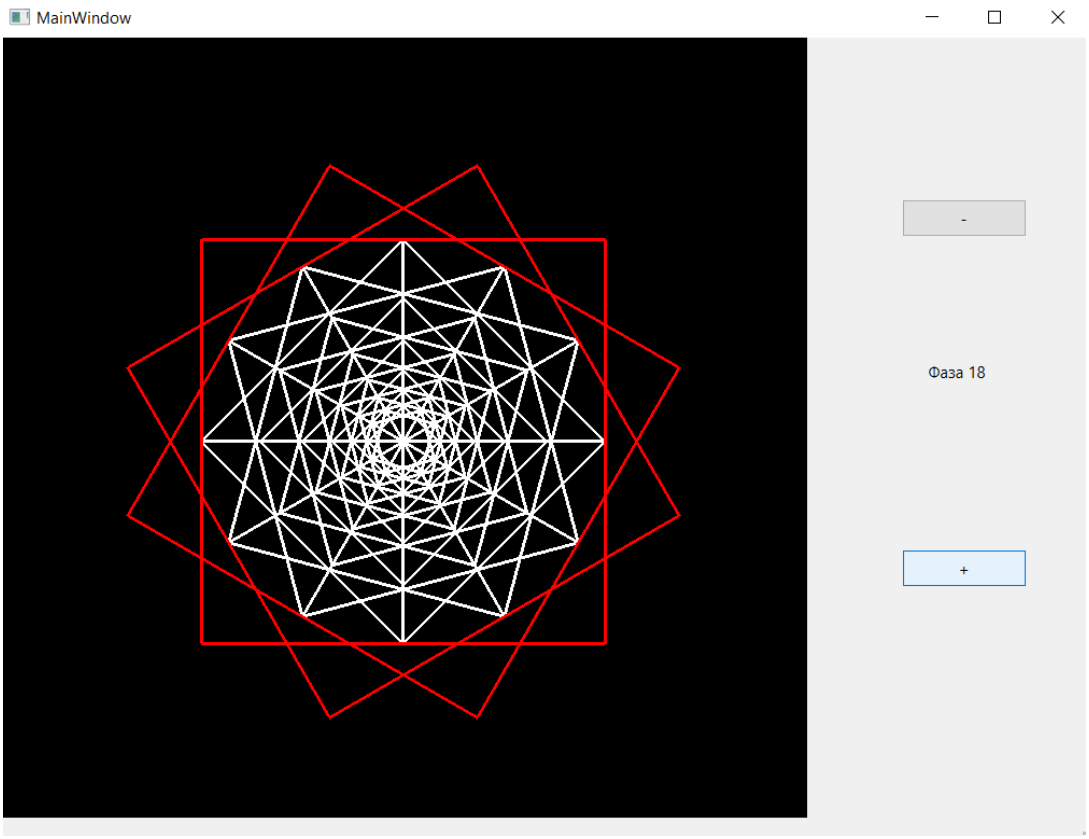


Рисунок 4 — 18 фаза

Большая фаза представлена на рисунке 5.

Рисунок 5 — 33 фаза

Вывод.

В результате выполнения лабораторной работы была разработана программа, отображающая фрактал. При выполнении работы были приобретены навыки работы с графической библиотекой OpenGL.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

main.py

```
import sys
import numpy as np
from PyQt6 import QtCore, QtWidgets
from OpenGL.GL import *
from OpenGL.GLU import *
from OpenGL.GLUT import *
from PyQt6.QtOpenGLWidgets import QOpenGLWidget
from PyQt6 import QtCore,QtWidgets

class MyGLWidget(QOpenGLWidget):
    def __init__(self,parent):
        super(MyGLWidget, self).__init__(parent)
        self.faze = 0
        self.coef = 1/np.sqrt(2)
        self.angle = 30
        self.figures = []
        self.base_polygon = np.array([
            [ 0. ,  0. ],
            [ 0. ,  0.5],
            [-0.5,  0. ],
            [ 0. , -0.5],
            [ 0.5,  0. ],
            [ 0. ,  0.5]
        ])

    def paintGL(self):
        self.fractal = []
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
        if self.faze > 0:
            self.fractal.append((self.base_polygon,"GL_TRIANGLE_FAN"))
            self.fractal_step()
        self.not_a_fractal()
        draw_fractal(self.fractal)

    def fractal_step(self):
        for i in range(self.faze - 1):
            new_figure = np.array([self.rotate_dots(dot,
np.deg2rad(self.angle)) for dot in self.fractal[i][0]])
            if (i + 1) % 3 == 0:
                self.fractal.append((new_figure *
self.coef,"GL_TRIANGLE_FAN"))
            else:
                self.fractal.append((new_figure,"GL_TRIANGLE_FAN"))

    def not_a_fractal(self):
        new_figure = np.array([
            [ 0.5,  0.5],
            [-0.5,  0.5],
            [-0.5, -0.5],
            [ 0.5, -0.5]
        ])
        for i in range(3):
            new_figure = np.array([self.rotate_dots(dot,
np.deg2rad(self.angle)) for dot in new_figure])
            self.fractal.append((new_figure,"GL_QUADS"))

    def rotate_dots(self, dot, angle):
        x_new = dot[0] * np.cos(angle) - dot[1] * np.sin(angle)
```

```
        y_new = dot[0] * np.sin(angle) + dot[1] * np.cos(angle)
        return [x_new, y_new]

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(810, 600)
        self.centralwidget = QtWidgets.QWidget(parent=MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.openGLWidget = MyGLWidget(parent=self.centralwidget)
        self.openGLWidget.setGeometry(QtCore.QRect(0, 0, 600, 600))
        self.openGLWidget.setObjectName("openGLWidget")
        self.minus = QtWidgets.QPushButton(parent=self.centralwidget)
        self.minus.setGeometry(QtCore.QRect(670, 120, 93, 28))
        self.minus.setObjectName("minus")
        self.minus.clicked.connect(self.press_minus)
        self.plus = QtWidgets.QPushButton(parent=self.centralwidget)
        self.plus.setGeometry(QtCore.QRect(670, 380, 93, 28))
        self.plus.setObjectName("plus")
        self.plus.clicked.connect(self.press_plus)
        self.label = QtWidgets.QLabel(parent=self.centralwidget)
        self.label.setGeometry(QtCore.QRect(690, 240, 55, 16))
        self.label.setObjectName("label")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(parent=MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 810, 26))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(parent=MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.minus.setText(_translate("MainWindow", "-"))
        self.plus.setText(_translate("MainWindow", "+"))
        self.label.setText(_translate("MainWindow", "Фаза 0"))

    def press_plus(self):
        self.openGLWidget.faze += 1
        self.label.setText("Фаза "+ str(self.openGLWidget.faze))
        self.openGLWidget.update()

    def press_minus(self):
        if self.openGLWidget.faze > 1:
            self.openGLWidget.faze -= 1
        self.label.setText("Фаза "+ str(self.openGLWidget.faze))
        self.openGLWidget.update()

func = {"GL_TRIANGLE_FAN" : GL_TRIANGLE_FAN, "GL_QUADS" : GL_QUADS}

def draw_fractal(fractal):
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glLineWidth(3)
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE)
    for f in fractal:
        glColor4f(1.0,1.0,1.0,1.0)
        if f[1] == "GL_QUADS":
            glColor4f(1.0, 0, 0,1.0)
        glBegin(func[f[1]])
```

```
        for i in range(len(f[0])):
            glVertex2f(f[0][i, 0], f[0][i, 1])
        glEnd()


if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec())
```