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

ОТЧЕТ

по лабораторной работе №4

по дисциплине «Компьютерная графика»

Тема: Построение фракталов

Студент гр.8382 _____ Нечепуренко Н.А.

Студент гр.8382 _____ Терехов А.Е.

Преподаватель _____ Герасимова Т.В.

Санкт-Петербург

2021

Цели работы.

На базе предыдущей лабораторной работы разработать программу реализующую фрактал по индивидуальному заданию. Требования и рекомендации к выполнению задания:

- проанализировать полученное задание, выделить информационные объекты и действия;
- разработать программу с использованием требуемых примитивов и атрибутов.

Задание.

Реализовать IFS-фрактал ”Ветка”(см. рис. 1).

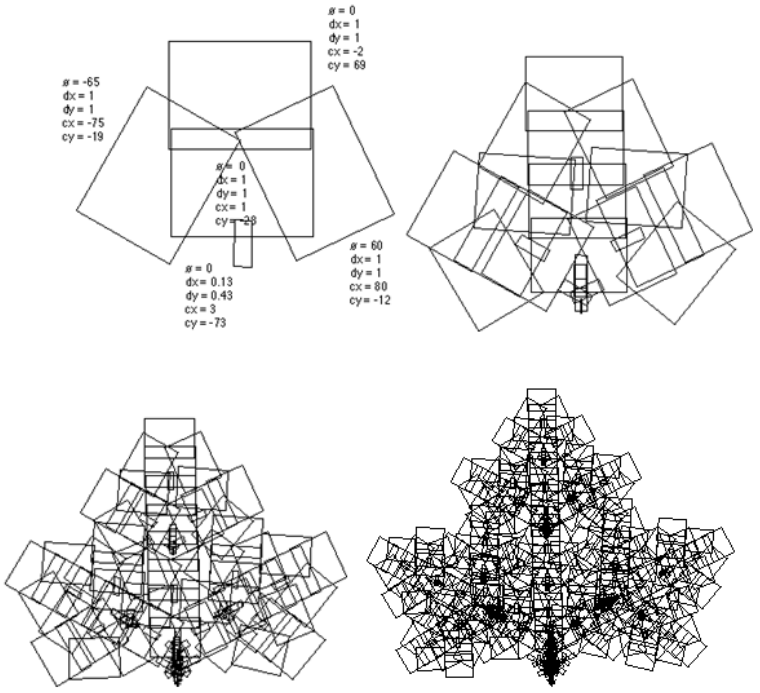

Рисунок 1 – Фрактал ”Ветка”пример построения

Основные теоретические положения.

IFS – Iterated Function System, система итерируемых функций, алгоритм фрактального кодирования, позволяющий сжимать данные благодаря использованию самоподобных участков. Эти функции представляют собой аффинные преобразования.

Суть метода заключается в следующем: пусть есть отображение компакта (X, d) в себя, т.е. $X \mapsto X$. Согласно теореме Банаха о фиксированной точке, существует единственная точка $\widetilde{x} \in X$, такая что, $w(\widetilde{x}) = \widetilde{x}$, она называется фиксированной точкой w. Более того $\forall x_0 \in X$ последовательность $x_{n+1} = w(x_n) \xrightarrow[n\to\infty]{} \widetilde{x}$.

Теперь представим, что имеется несколько подобных отображений w_i на X. У каждого из них есть своя фиксированная точка $\widetilde{x_i}$. Оказывается, что если выбирать w_i случайно на каждой итерации, каждое отображение будет стараться ”притянуть”точку к своей фиксированной. Если сопоставить каждому отображению w_i какую-то вероятность $p_i : \sum_i p_i = 1$, то при так называемом случайном блуждании будет получен необходимый фрактал. Подробнее можно прочитать по этой ссылке.

Выполнение работы.

Заметим, что в приведенном фрактале (см. рис. 1) имеется 5 генераторов. Пронумеруем их от 1 до 5.

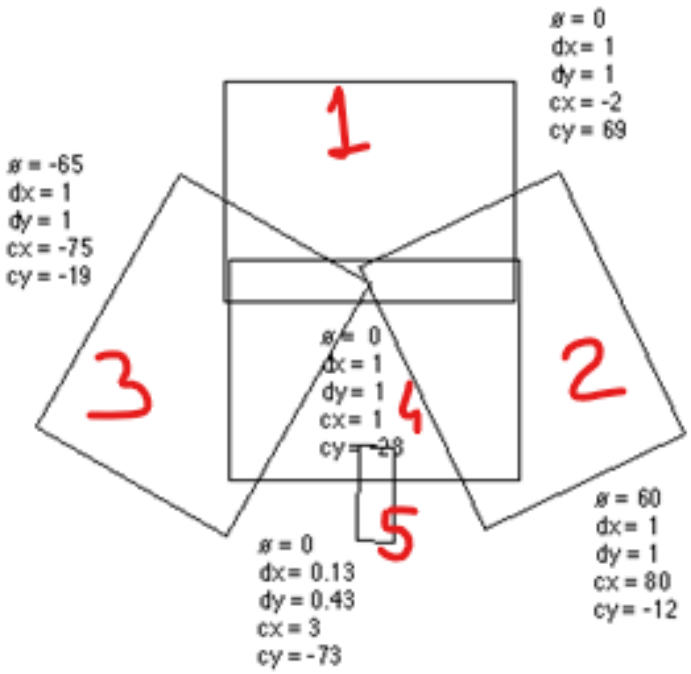

Рисунок 2 – Основные генераторы фрактала

Пусть центр каждого генератора будет фиксированной точкой, в терминах IFS (см. выше), тогда отображения будут иметь вид:

$$f_1(\begin{pmatrix} x \\ y \end{pmatrix}) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -0.02 \\ 0.69 \end{pmatrix}$$

$$f_2(\begin{pmatrix} x \\ y \end{pmatrix}) = \begin{pmatrix} \frac{1}{2} & -\frac{\sqrt{3}}{2} \\ \frac{\sqrt{3}}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0.8 \\ -0.12 \end{pmatrix}$$

$$f_3(\begin{pmatrix} x \\ y \end{pmatrix}) = \begin{pmatrix} 0.423 & -0.906 \\ 0.906 & 0.423 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -0.75 \\ -0.19 \end{pmatrix}$$

$$f_4(\begin{pmatrix} x \\ y \end{pmatrix}) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0.01 \\ -0.28 \end{pmatrix}$$

$$f_5(\begin{pmatrix} x \\ y \end{pmatrix}) = \begin{pmatrix} 0.13 & 0 \\ 0 & 0.43 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 0.03 \\ -0.73 \end{pmatrix}$$

Отображения равновероятны. При использовании приведенных отображений результатом случайного блуждания будет требуемый фрактал.

Для выполнения работы был выбран язык Python 3.8 с библиотеками PyQt5 и PyOpenGL. Для их установки необходимо воспользоваться командами:

```
pip install pyqt5 PyOpenGL PyOpenGL_accelerate
```

Запуск программы:

```
python3 main.py
```

В программе отображения имеют вид:

```
    leaf_params = [
    [1., 0., 0., 1., -.02, .69],
    [1 / 2, -math.sqrt(3) / 2, math.sqrt(3) / 2, 1 / 2, .8,
       -.12],
    [0.423, 0.906, -0.906, 0.423, -.75, -.19],
    [1, 0, 0, 1, .01, -.28],
    [.13, 0, 0, .43, .03, -.73]
]
    ...
    self._function_generator = lambda a, b, c, d, e, f: lambda x0
       , y0: (a * x0 + b * y0 + e, c * x0 + d * y0 + f)
    self._functions = [self._function_generator(*params) for
       params in params_list]
```

Процесс случайного блуждания:

```
    def paint(self, num_iterations=100000, color_mode=False):
      current_x, current_y = self._x, self._y
      glBegin(GL_POINTS)
      glColor3f(1., 0., 0.)
      for i in range(num_iterations):
          color_index, (current_x, current_y) = self.
             _fractal_generator.getNext(current_x, current_y)
```

```
            if color_mode:
                glColor3f(*color_params[color_index])
            glVertex2f(current_x / 2, current_y / 2)
        glEnd()
```

Результаты программы на 1000 и 100000 приведены на рисунках 3 и 4 соответственно.

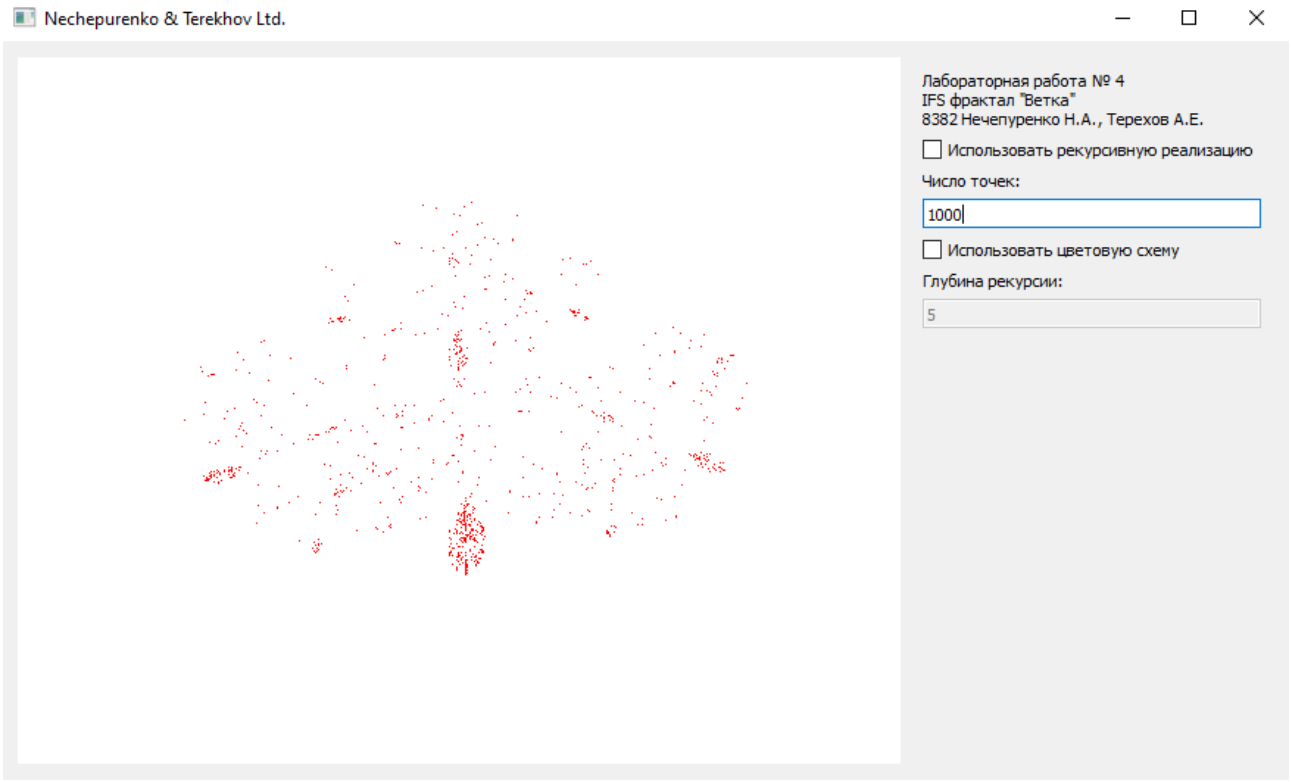

Рисунок 3 – Случайное блуждание при 1000 итерациях

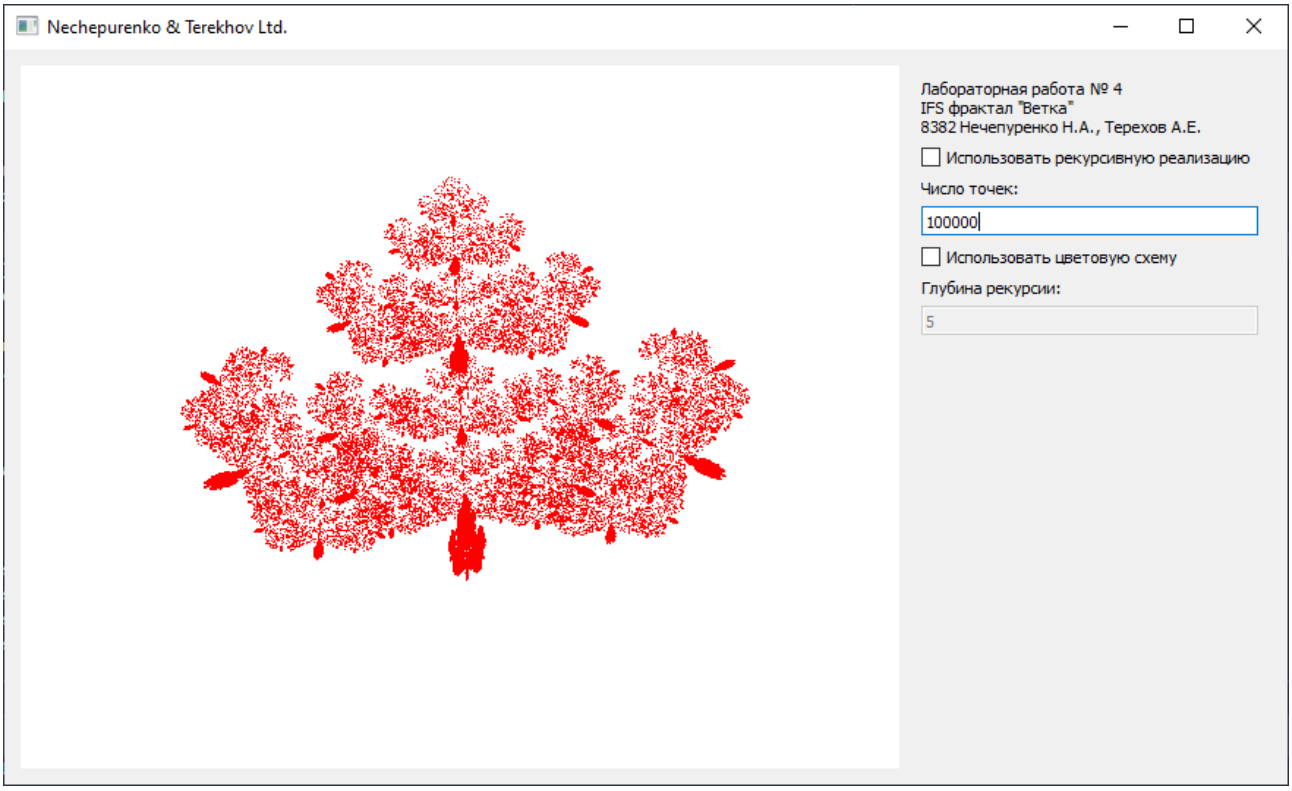

Рисунок 4 – Случайное блуждание при 100000 итерациях

Также было реализовано построение фрактала с помощью боксов и рекурсивной функции.

```
...
rot = [
    [math.cos, lambda theta: -math.sin(theta)],
    [math.sin, math.cos],
]

def __init__(self):
    self._start_shape = [
        [-1, 1],
        [1, 1],
        [1, -1],
        [-1, -1],
    ]
```

```
    self._sqs = []
    self._max_depth = 5
    self._params = [
        [0, 1, 1, 0, -.4],
        [0, 1, 1, 0, .4],
        [- 60 * math.pi / 180, 1, 1, -.5, -.2],
        [60 * math.pi / 180, 1, 1, .5, -.2],
        [0, .1, .5, 0, -2.4],
    ]
...
def get(self, depth):
    self._max_depth = depth
    self._sqs = []
    for index in range(5):
        self.rec(self._start_shape, *self._params[index])
    return list(filter(lambda x: x is not None, self._sqs))
...
def rec(self, shape: list, theta: float = 0, dx: float = 1,
   dy: float = 1, cx: float = 0, cy: float = 0,
        depth=0):
    new_shape = []
    for s in shape:
        new_shape.append(
            [(self.rot[0][0](theta) * s[0] + cx) * dx * .5 +
               (self.rot[0][1](theta) * s[1] + cx) * dy * .5,
             (self.rot[1][0](theta) * s[0] + cy) * dx * .5 +
                (self.rot[1][1](theta) * s[1] + cy) * dy *
                .5])
    if depth == self._max_depth - 1:
        for index in range(5):
            self._sqs.append(self.rec(new_shape, *self.
               _params[index], depth + 1))
```

```
        elif depth < self._max_depth:
            for index in range(5):
                self.rec(new_shape, *self._params[index], depth +
                    1)
        return new_shape
```

Результаты построения фрактала с глубиной 3 и 5 приведены на рисунках 5 и 6.

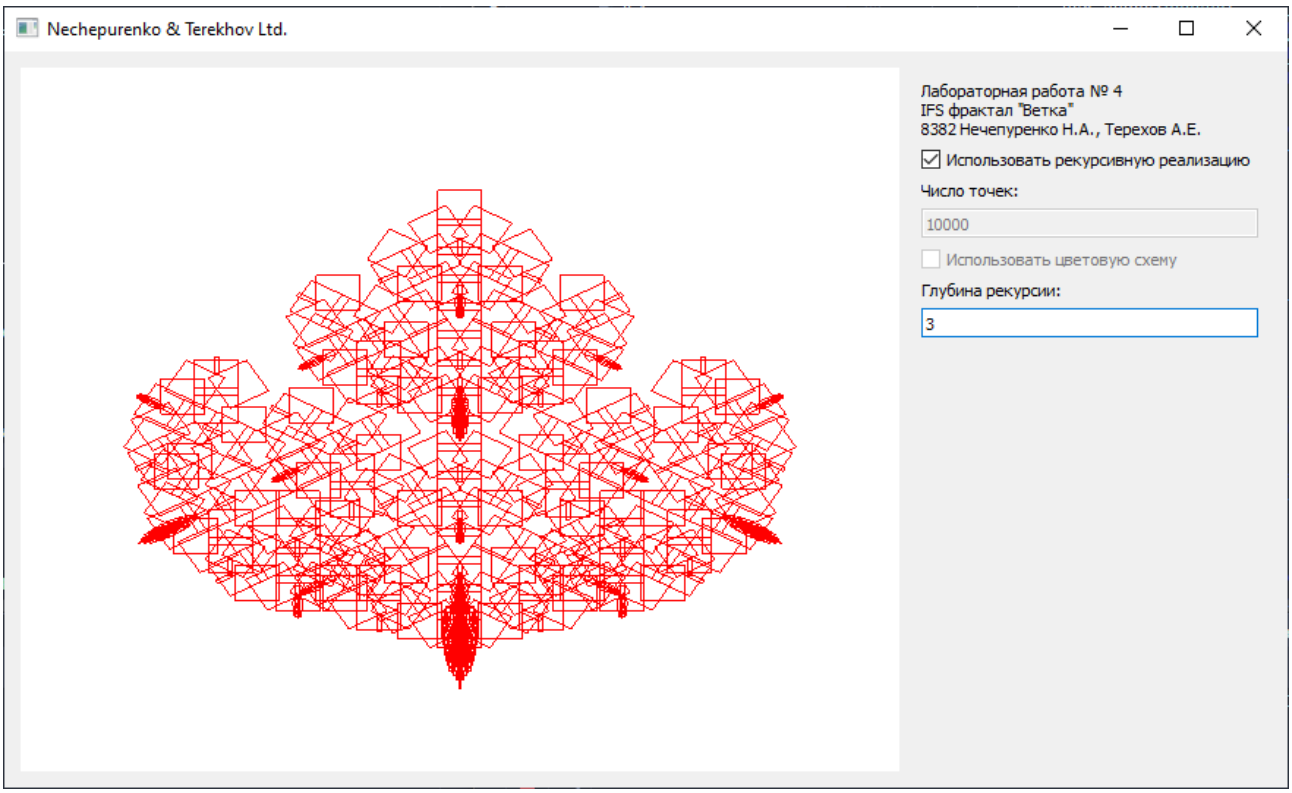

Рисунок 5 – Рекурсия с глубиной 3

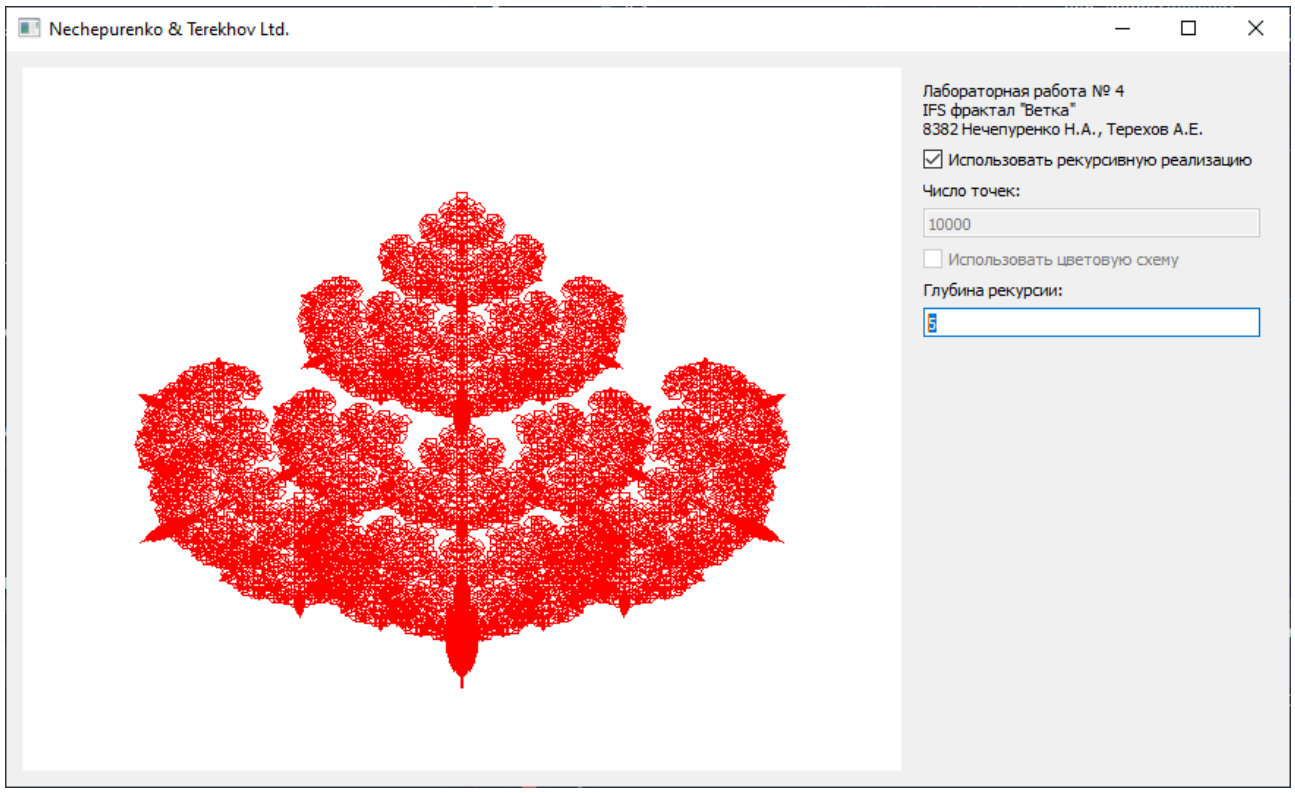

Рисунок 6 – Рекурсия с глубиной 5

С полным исходным кодом программы можно ознакомиться по ссылке.

Выводы.

В ходе выполнения лабораторной работы были получены навыки работы с фракталами, их построение средствами библиотеки OpenGL. Был изучен подход к построению фракталов с помощью IFS, а также реализован наивный рекурсивный алгоритм построения заданного объекта.