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

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Компьютерная графика»
Тема: Построение фракталов.

Студент гр. 0304 _____ Алексеев Р.В.

Преподаватель _____ Герасимова Т.В.

Санкт-Петербург
2023

Цель работы.

- Освноение работы с фракталами.

Задание.

На базе предыдущей лабораторной работы разработать программу реализующую фрактал по индивидуальному заданию.

Выполнение работы.

Для создания фрактала была создана функция *void createStepFractal(float coeff, bool rotate).* Реализация этой функции представлена далее.

Код функции *createStepFractal*:

void GLWidget::createStepFractal(float coeff, bool rotate)

```
{
            QColor colors[3] = {QColor("black"), QColor("lightGreen"), QColor("darkGreen")};
```

```
        QColor colorsLine[3] = {QColor("lightYellow"), QColor("darkRed"),
QColor("yellow")};

    float a = rotate ? startAngle * coeffRotateAngle : startAngle;
    float angleIncrement = 2.0f * M_PI / 5;
    float edge = radius * coeff;

    for(int i = 0; i < 5; i++)
    {

        // Painting triangles
        glBegin(GL_TRIANGLES);
                            glColor3d(colors[2].redF(),  colors[2].greenF(),
colors[2].blueF());
            glVertex2d(edge*cos(a), edge*sin(a));
                            glColor3d(colors[1].redF(),  colors[1].greenF(),
colors[1].blueF());
              glVertex2d(edge*cos(a) * coeffIncreaseEdge * coeffIncreaseEdge,
edge*sin(a) * coeffIncreaseEdge * coeffIncreaseEdge);
                            glColor3d(colors[2].redF(),  colors[2].greenF(),
colors[2].blueF());
             glVertex2d(-edge*cos(a + angleIncrement*2) * coeffIncreaseEdge, -
edge*sin(a + angleIncrement*2) * coeffIncreaseEdge);
        glEnd();

        // Painting lines around triangles
        glBegin(GL_LINE_LOOP);
                    glColor3d(colorsLine[0].redF(),  colorsLine[0].greenF(),
colorsLine[0].blueF());
            glVertex2d(edge*cos(a), edge*sin(a));
                    glColor3d(colorsLine[0].redF(),  colorsLine[0].greenF(),
colorsLine[0].blueF());
              glVertex2d(edge*cos(a) * coeffIncreaseEdge * coeffIncreaseEdge,
edge*sin(a) * coeffIncreaseEdge * coeffIncreaseEdge);
                    glColor3d(colorsLine[0].redF(),  colorsLine[0].greenF(),
colorsLine[0].blueF());
             glVertex2d(-edge*cos(a + angleIncrement*2) * coeffIncreaseEdge, -
edge*sin(a + angleIncrement*2) * coeffIncreaseEdge);
        glEnd();

        glBegin(GL_LINE_LOOP);
```

```
                    glColor3d(colorsLine[0].redF(),  colorsLine[0].greenF(),
colorsLine[0].blueF());
            glVertex2d(-edge*cos(a + angleIncrement*2) * coeffIncreaseEdge, -
edge*sin(a + angleIncrement*2) * coeffIncreaseEdge);
                    glColor3d(colorsLine[0].redF(),  colorsLine[0].greenF(),
colorsLine[0].blueF());
                    glVertex2d(edge*cos(a  -  angleIncrement),  edge*sin(a  -
angleIncrement));
                    glColor3d(colorsLine[0].redF(),  colorsLine[0].greenF(),
colorsLine[0].blueF());
               glVertex2d(edge*cos(a - angleIncrement) * coeffIncreaseEdge *
coeffIncreaseEdge,  edge*sin(a  -  angleIncrement)  *  coeffIncreaseEdge  *
coeffIncreaseEdge);
        glEnd();

        // Painting spirals
        float x = edge*cos(a) * coeffIncreaseEdge * coeffIncreaseEdge;
        float y = edge*sin(a) * coeffIncreaseEdge * coeffIncreaseEdge;
        float sx = -edge*cos(a + angleIncrement*2) * coeffIncreaseEdge;
        float sy = -edge*sin(a + angleIncrement*2) * coeffIncreaseEdge;
        float dx = (sx-x)/10;
        float dy = (sy-y)/10;

        float dvx = (edge*cos(a) + x)/200;
        float dvy = (edge*sin(a) + y)/200;

        glLineWidth(2);

        glBegin(GL_LINE_STRIP);

            for(int e = 0.; e < 5; e++)
            {
                    glColor3d(colorsLine[2].redF(),  colorsLine[2].greenF(),
colorsLine[2].blueF());
                glVertex2d(x, y);
                x += dx + dvx;
                y += dy + dvy;

                dvx *= 0.8;
                dvy *= 0.8;
            }
```

```
            for(int e = 0; e < 6; e++)
            {
                      glColor3d(colorsLine[2].redF(), colorsLine[2].greenF(),
colorsLine[2].blueF());
                glVertex2d(x, y);
                x += dx - dvx;
                y += dy - dvy;

                dvx /= 0.8;
                dvy /= 0.8;
            }

        glEnd();

        glLineWidth(1);

        a += angleIncrement;
    }
}
```

Данная функция вызывается в цикле, который определяет глубину фрактала. Функция принимает коэффициент уменьшения размеров — coeff, коэффициент поворота — rotate.

Функция сначала отрисовывает зеленый и черный треугольники, а также их контуры светожелтого цвета, по часовой стрелке при четных итерациях и против при нечетных, т. к. в зависимости от четности итерации получаемый пятиугольник переворачивается. Также отрисовывается фрагмент спирали в зеленом треугольнике. Цикл повторяет этот шаг 5 раз, тем самым получается пятиугольник из закрашенных треугольников.

Внешний цикл, вызывающий данную функцию, уменьшает coeff с каждой иттерацией, тем самым уменьшается размер нового шага фрактала.

Для выбора глубины фрактала был добавлен элемент в интерфейс программы.

Тестирование.

Программа была протестирована с разными заданными глубинами фрактала. Результаты представлены на рис. 1 — 3.

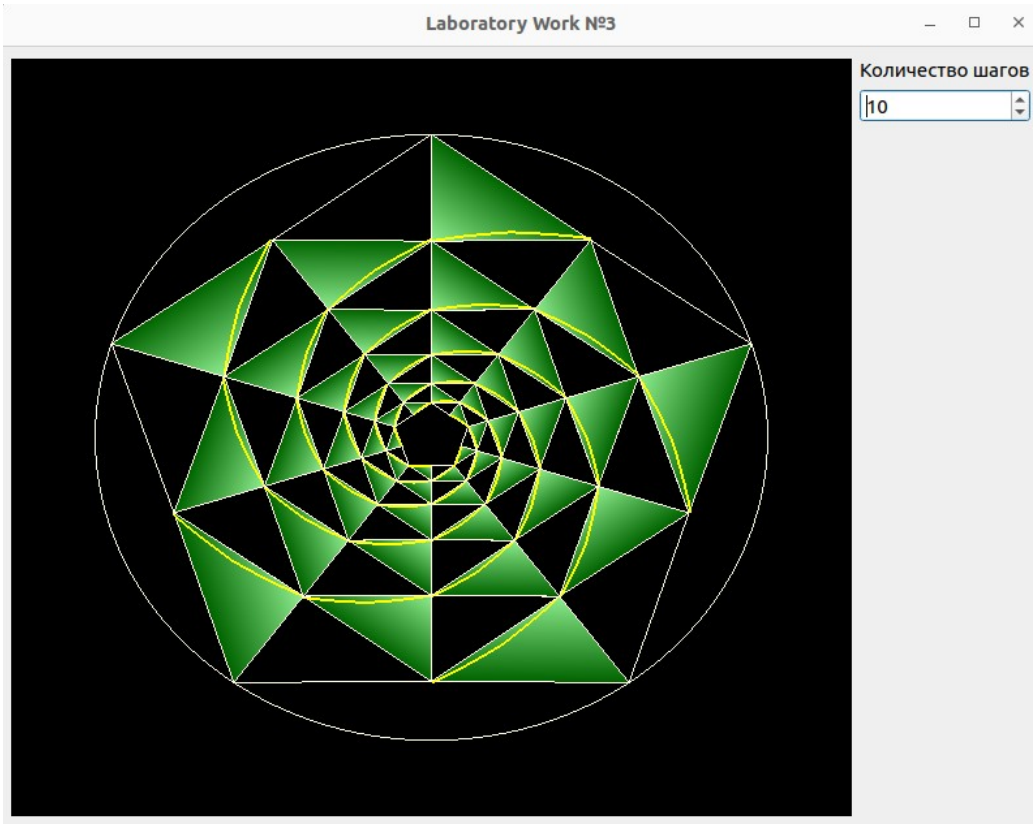

Рисунок 1 — Глубина фрактала — 10.

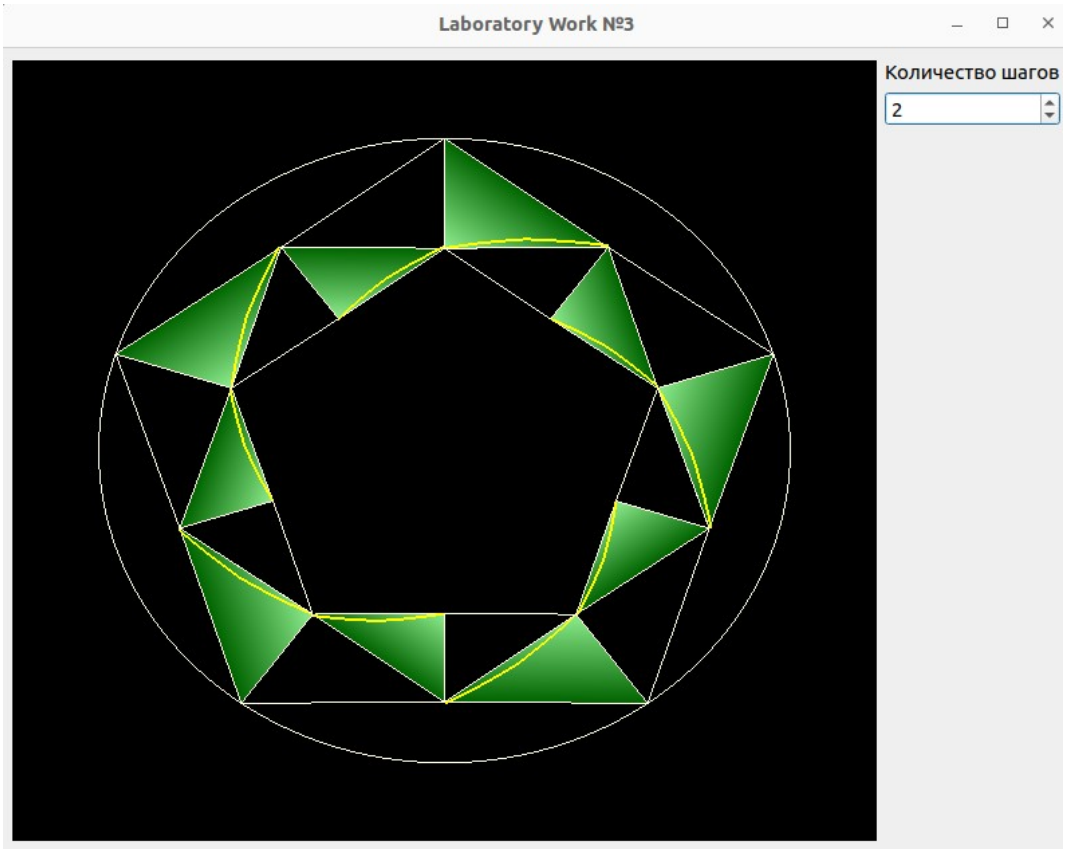

Рисунок 2 — Глубина фрактала — 2.

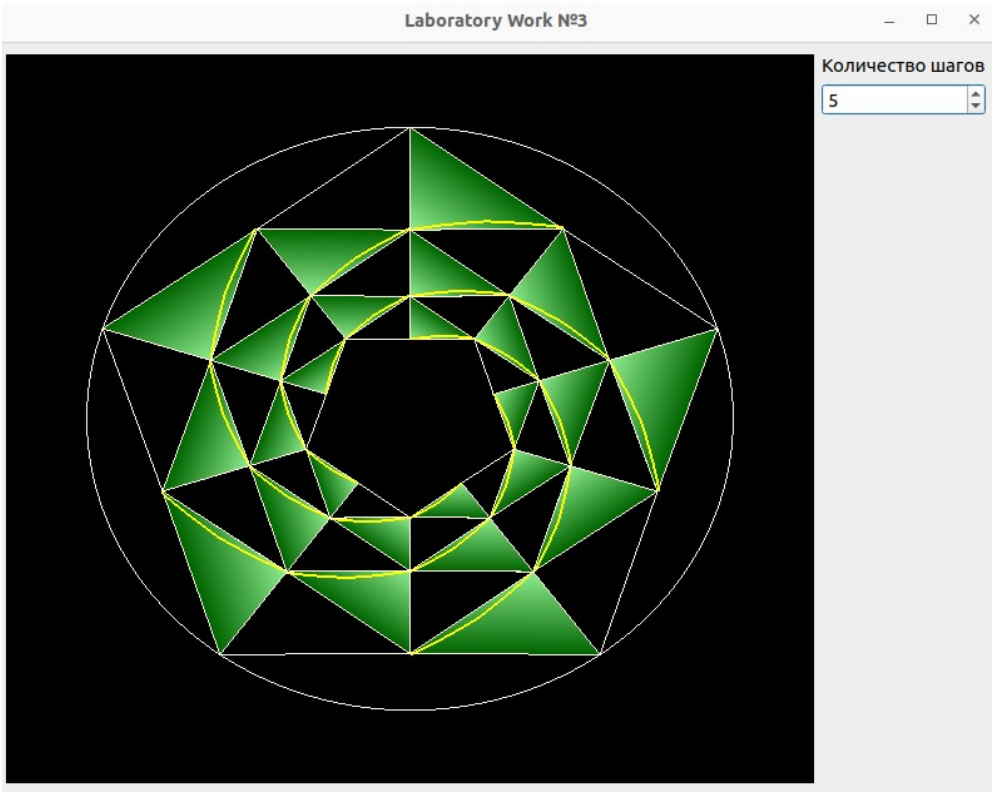

Рисунок 3 — Глубина фрактала — 5.

Выводы.

В ходе работы на основе программы из прошлых лабораторных работ была создана программа, отрисовывающая заданный фрактал.