Московский авиационный институт
(Национальный исследовательский университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

Лабораторные работы №4, 5

по курсу «Компьютерная графика»
Тема: «Ознакомление с технологией OpenGL»

Студент: Мариничев И. А.
Группа: М8О-308Б-19
Преподаватель: Филиппов Г. С.
Оценка:

Москва
2021

1. Постановка задачи.

Создать графическое приложение с использованием OpenGL. Используя результаты Л.Р.№3, изобразить заданное тело (то же, что и в л.р. №3) с использованием средств OpenGL 2.1. Использовать буфер вершин. Точность аппроксимации тела задается пользователем. Обеспечить возможность вращения и масштабирования многогранника и удаление невидимых линий и поверхностей. Реализовать простую модель освещения на GLSL. Параметры освещения и отражающие свойства материала задаются пользователем в диалоговом режиме.

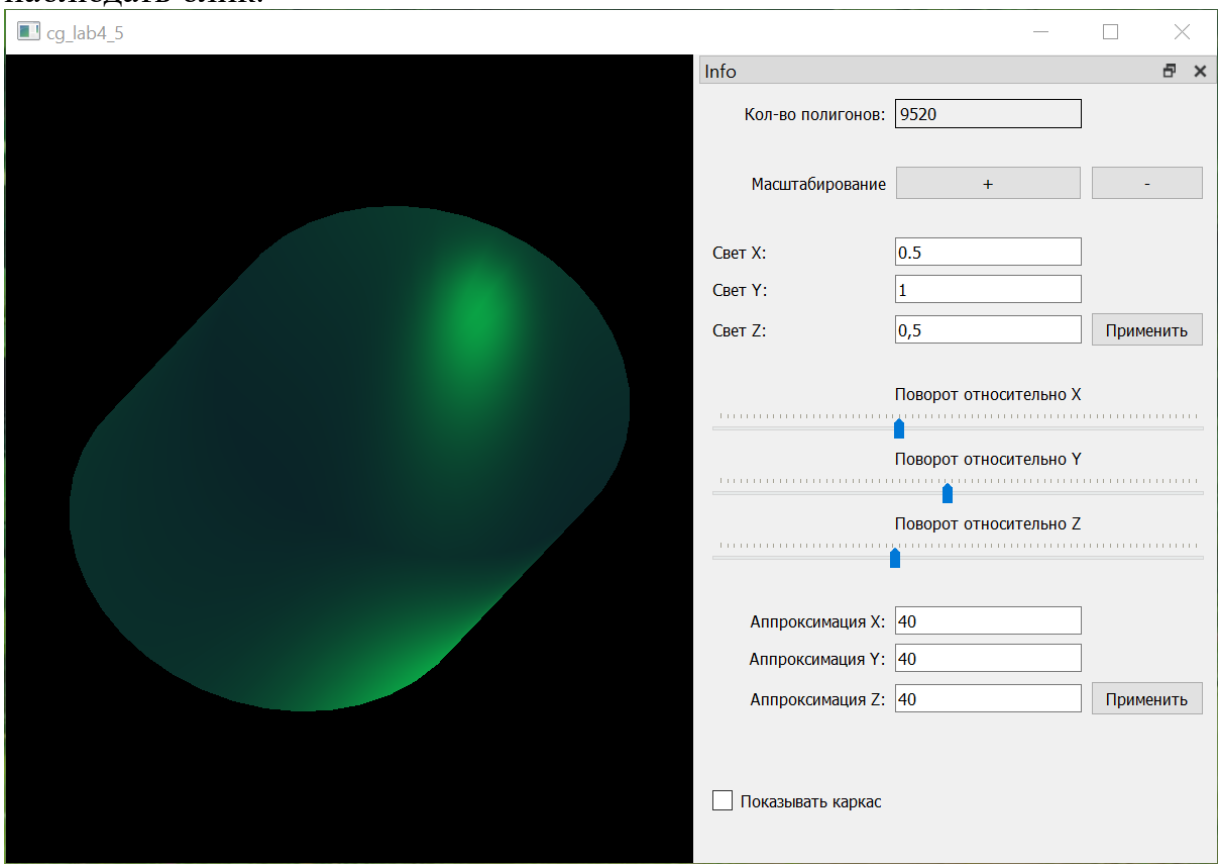
Вариант №8: Наклонный круговой цилиндр.

2. Описание программы.

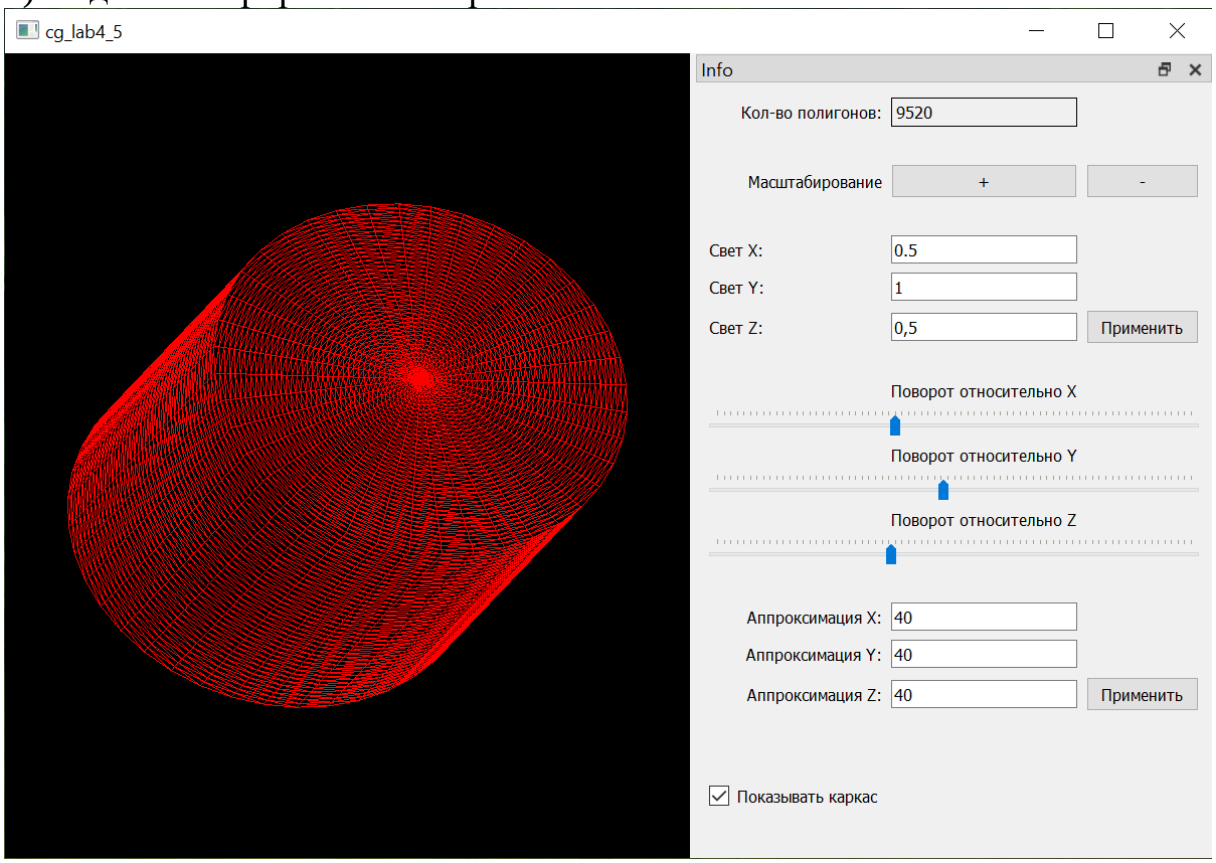
Фигура строится по аналогии с предыдущей лабораторной работой, только при помощи средств OpenGL. Здесь так же присутствует класс `polygon` для хранения полигонов, класс `oblique_circular_cylinder`, представляющий фигуру наклонный круговой цилиндр. Такая фигура состоит из множества полигонов. Пользователь может задавать аппроксимацию тела по разным осям. Все преобразования для фигуры выполняются средствами OpenGL.

3. Демонстрация работы программы.

1) Вид тела при средней аппроксимации по всем осям. На теле можно наблюдать блик.



2) Вид тела с прорисовкой каркаса.



4. Основной код программы.

```
void display::initializeGL() {
    initializeOpenGLFunctions();
    glClearColor(0.f, 0.f, 0.f, 1.f);
    glEnable(GL_DEPTH_TEST);
}

void display::resizeGL(int width, int height) {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glViewport(0, 0, width, height);
}

void display::paintGL() {
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_DEPTH_TEST);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();

    glRotatef(rotateX, 1.f, 0.f, 0.f);
    glRotatef(rotateY, 0.f, 1.f, 0.f);
    glRotatef(rotateZ, 0.f, 0.f, 1.f);
    glScalef(scale, scale, scale);

    if (displayCarcass) {
        glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
        glDisable(GL_LIGHTING);
    } else {
```

```
        glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
        glEnable(GL_LIGHTING);
    }

    glPushMatrix();
    glLoadIdentity();
    glEnable(GL_NORMALIZE);
    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, fig.get_ambient_color());
    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, fig.get_diffuse_color());
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, fig.get_specular_color());
    glMaterialf(GL_FRONT_AND_BACK, GL_SHININESS, fig.get_shininess());

    float light_ambient[] = {0.f, 0.22f, 0.51f, 1.f};
    float light_diffuse[] = {0.f, 0.55f, 0.128f, 1.f};
    float light_specular[] = {0.f, 0.44f, 0.102f, 1.f};
    float light_position[] = {lightPositionX,
                              lightPositionY,
                              lightPositionZ, 1.f};

    glEnable(GL_LIGHT0);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
    glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glLightf(GL_LIGHT0, GL_SPOT_EXPONENT, 128);
    glLightf(GL_LIGHT0, GL_CONSTANT_ATTENUATION, 1.f);
    glPopMatrix();
    glEnable(GL_CULL_FACE);
    glCullFace(GL_FRONT);
    glColor3f(1.f, 0.f, 0.f);
    for (auto polygon: fig.get_polygons()) {
        glBegin(GL_POLYGON);
            for (auto vertex: polygon.vertices) {
                glVertex3f(vertex.x(), vertex.y(), vertex.z());
            }
        glEnd();
    }
    glDisable(GL_CULL_FACE);
    glDisable(GL_LIGHT0);
    glDisable(GL_LIGHTING);
    glDisable(GL_DEPTH_TEST);
}
```

5. Выводы.

В ходе данной лабораторной работы я получил базовые навыки работы с OpenGL.