Московский авиационный институт
(Национальный исследовательский университет)

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

Лабораторная работа №6

по курсу «Компьютерная графика»
Тема: «Создание шейдерных анимационных эффектов в OpenGL 2.1»

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

Москва
2021

1. Постановка задачи.

Для поверхности, созданной в л.р. №5, обеспечить выполнение следующего шейдерного эффекта:

Вариант №8: Прозрачность вершины обратно пропорциональна расстоянию от заданной точки.

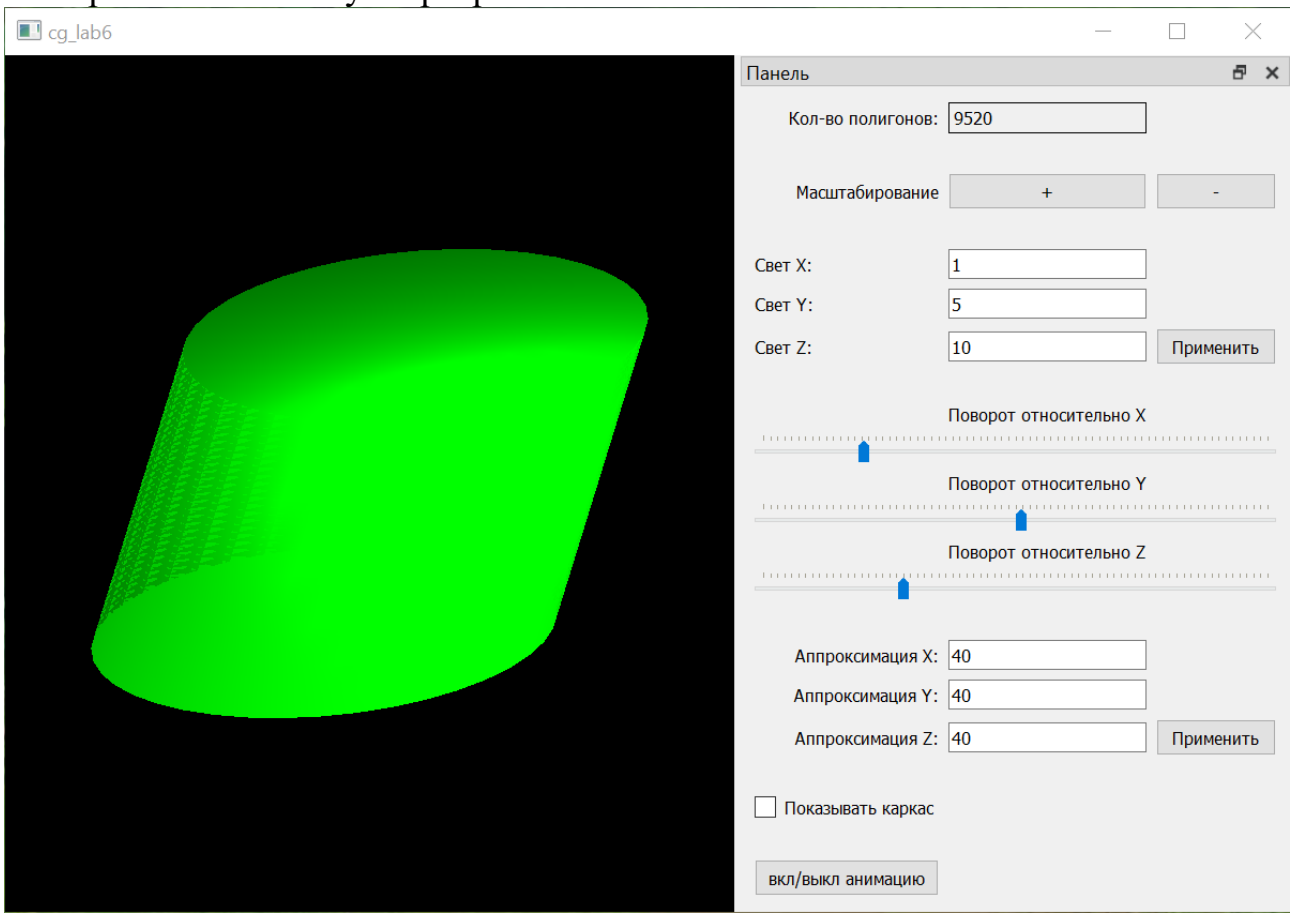
2. Описание программы.

Фигура строится 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.

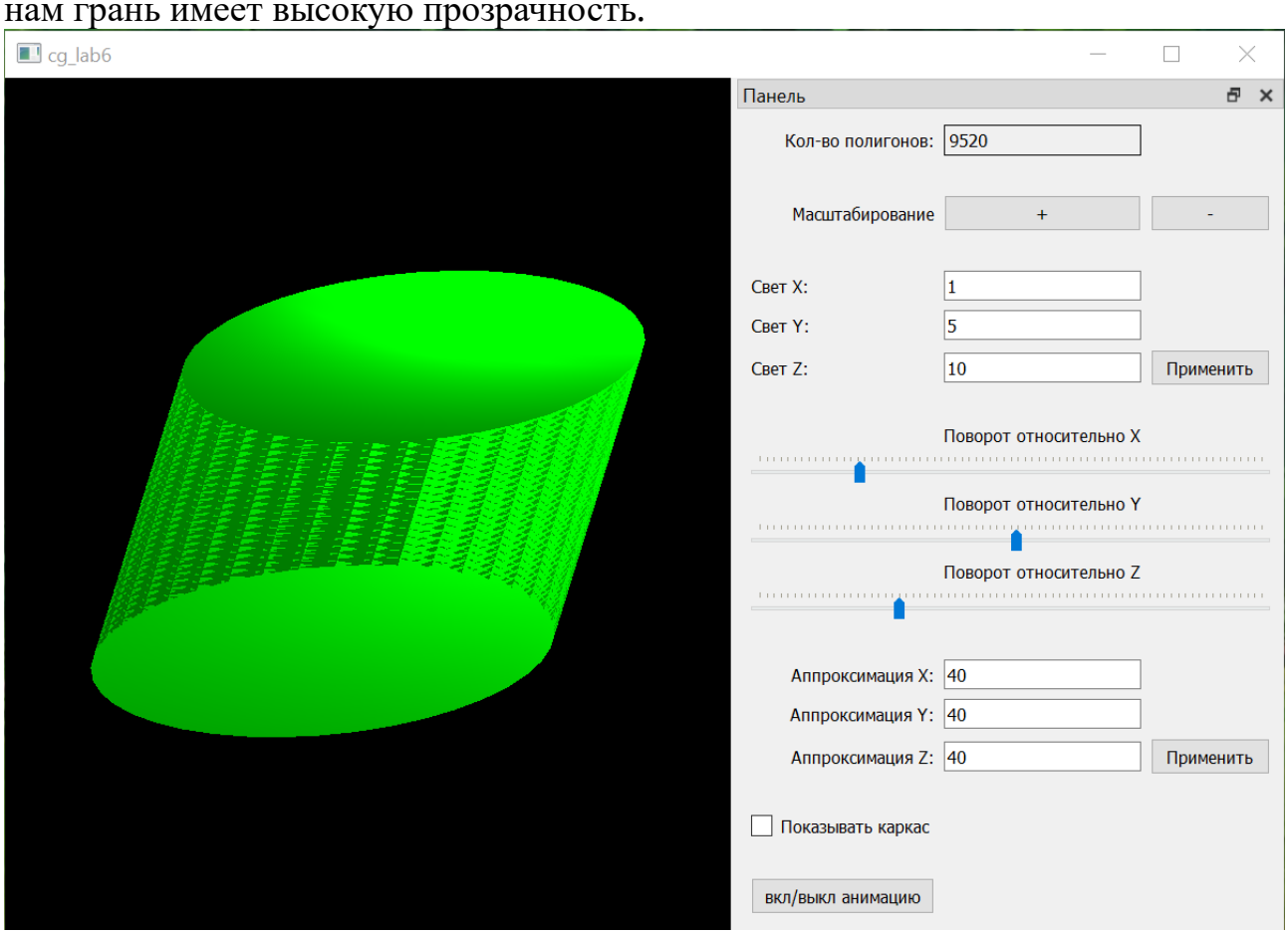
Для демонстрации изменения прозрачности вершин я реализовал таймер, который задаёт угол для уравнения, по которому вращается точка. От каждой точки цилиндра измеряется расстояние до вращающейся точки и обратно пропорционально изменяется прозрачность.

3. Демонстрация работы программы.

1) Вид тела, когда вращающаяся точка находится за цилиндром, а ближняя к нам грань имеет низкую прозрачность.



2) Вид тела, когда вращающаяся точка находится перед цилиндром, а ближняя к нам грань имеет высокую прозрачность.



4. Основной код программы.

```
void display::anim_draw() {
    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);
    }

    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);

    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);

    float distance;
    float x1 = 15 * cos(time),
          y1 = 15 * sin(time),
          z1 = 5;

    for (auto polygon: fig.get_polygons()) {
        glBegin(GL_POLYGON);
            for (auto vertex: polygon.vertices) {
                distance = std::sqrtf((vertex.x() - x1) * (vertex.x() - x1)
                                      + (vertex.y() - y1) * (vertex.y() - y1)
                                      + (vertex.z() - z1) * (vertex.z() - z1));
                glColor4f(0.f,  153.0f,  0.0f,  12.f  *  static_cast<float>(1  /
distance));

                glVertex3f(vertex.x(),
                           vertex.y(),
                           vertex.z());
            }
        glEnd();
    }
    glDisable(GL_BLEND);
    glDisable(GL_LIGHT0);
    glDisable(GL_LIGHTING);
}

void MainWindow::on_off_animation() {
    if (!animation) {
        display_ptr->start_timer();
        animation = true;
        display_ptr->glDisable(GL_CULL_FACE);
        display_ptr->glEnable(GL_COLOR_MATERIAL);
        display_ptr->updateGL();
    } else {
        display_ptr->stop_timer();
        animation = false;
        display_ptr->glEnable(GL_CULL_FACE);
        display_ptr->glDisable(GL_COLOR_MATERIAL);
        display_ptr->updateGL();
    }
}
```

5. Выводы.

В ходе данной лабораторной работы я научился реализовывать шейдерные анимации на OpenGL.