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

# ОТЧЕТ

**по лабораторной работе №6**

**по дисциплине «Компьютерная графика»**

**Тема: Реализация трехмерного объекта с использованием библиотеки OpenGL.**

Студент гр. 8303 _____ Кибардин А.Б.

Студент гр. 8303 _____ Крыжановский К.Е.

Преподаватель _____ Герасимова Т.В.

Санкт-Петербург

2021

**Цель работы**

Ознакомление принципов работы с шейдерами в OpenGL.

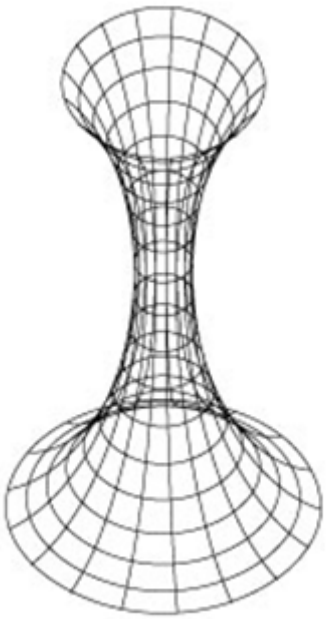
**Задание**

Разработать программу, реализующую представление разработанного вами трехмерного рисунка, используя предложенные функции библиотеки OpenGL (матрицы видового преобразования, проецирование) и язык GLSL.

Разработанная программа должна быть пополнена возможностями остановки интерактивно различных атрибутов через вызов соответствующих элементов интерфейса пользователя, замена типа проекции, управление преобразованиями, как с помощью мыши, так и с помощью диалоговых элементов.

**Основные теоретические положения**

**Вариант 45**



**Ход выполнения работы**

1. Была написана программа для рисования каркасного объекта.

2. Для изображения заданного каркасного объекта был написан метод draw для виджета класса GLWidget, являющегося оберткой над функциями OpenGL

Код метода:

```
void GLWidget::draw() {
   float u;
   GLfloat vec[countCircles * 3 ];
     GLfloat totalPoints[(countSteps * 3 * countCircles + countSteps * 3 * (countCircles - 2))];
   int j = 0;
   //круги
   for (double t = 0; t < 2 * M_PI; t += 2.0 * M_PI / countSteps / 100) {
       m_program->setUniformValue(m_colAttr, t / 2 / M_PI, t / 2 / M_PI, 1.0f, 1.0f);
      for (int i = 0; i < countCircles; i += 1) {
         if (i != 0){
            if (i > countCircles){
               u = -log(i - countCircles);
            } else {
               u = log(i);
            }
         } else {
            u = 0;
         }
         vec[3 * i] = x0 + R * cosh(u) * cos(t);
         vec[3 * i + 1] = y0 + R * sinh(u) * sin(t);
         vec[3 * i + 2] = z0 + R * sinh(u);
      }

      glVertexAttribPointer(m_posAttr, 3, GL_FLOAT, GL_FALSE, 0, vec);
      glEnableVertexAttribArray(m_posAttr);
      glDrawArrays(GL_POINTS, 0, countCircles);
      glDisableVertexAttribArray(m_posAttr);
   }

   //линии
   j = 0;
   for (double t = 0; t < 2 * M_PI; t += 2.0 * M_PI / countSteps) {
```

```
        m_program->setUniformValue(m_colAttr, t / 2 / M_PI, t / 2 / M_PI, 1.0f,
1.0f);
       // Работает
       for (int i = 0; i < countCircles; i += 1) {
          if (i != 0){
             if (i > countCircles){
                u = -log(i - countCircles);
             } else {
                u = log(i);
             }
          } else {
             u = 0;
          }
          vec[3 * i] = x0 + R * cosh(u) * cos(t);
          vec[3 * i + 1] = y0 + R * sinh(u) * sin(t);
          vec[3 * i + 2] = z0 + R * sinh(u);

          totalPoints[j] = vec[3 * i];
          totalPoints[j + 1] = vec[3 * i + 1];
          totalPoints[j + 2] = vec[3 * i + 2];
          j += 3;
          if (i != 0 && i != countCircles - 1){
             totalPoints[j] = vec[3 * i];
             totalPoints[j + 1] = vec[3 * i + 1];
             totalPoints[j + 2] = vec[3 * i + 2];
             j += 3;
          }
       }
    }

          glVertexAttribPointer(m_posAttr, 3, GL_FLOAT, GL_FALSE, 0,
totalPoints);
    glEnableVertexAttribArray(m_posAttr);
      glDrawArrays(GL_LINES, 0, countSteps * countCircles + countSteps *
(countCircles - 2));
    glDisableVertexAttribArray(m_posAttr);
    m_program->setUniformValue(m_colAttr, 0.5f, 0.7f, 1.0f, 1.0f);

    m_program->release();
```

```
}
```

3. Для смены ортогонального и перспективного отображения был написан метод changeProection

```
void Widget::changeProection() {
    proection = !proection;
    m_projectionMatrix.setToIdentity();
    if (proection) {
        m_projectionMatrix.ortho(-12.0f,12.0f,-12.0f,12.0f,0.1f,100.0f);
    } else {
        m_projectionMatrix.perspective(45.0f, 5, 0.01f, 100.0f);
    }
    this->update();
}
```

4. Код вершинного шейдера

```
attribute highp vec4 a_position;
attribute highp vec2 a_texcoord;
attribute highp vec3 a_normal;

uniform highp mat4 u_projectionMatrix;
uniform highp mat4 u_viewMatrix;
uniform highp mat4 u_modelMatrix;

varying highp vec2 v_texcoord;
varying highp vec3 v_normal;
varying highp vec4 v_position;

void main(void)
{
    mat4 mv_matrix = u_viewMatrix * u_modelMatrix;


    gl_Position = u_projectionMatrix * mv_matrix * a_position;

    v_texcoord = a_texcoord;

    v_normal = normalize(vec3(mv_matrix * vec4(a_normal, 0.0)));
    v_position = mv_matrix * a_position;
```

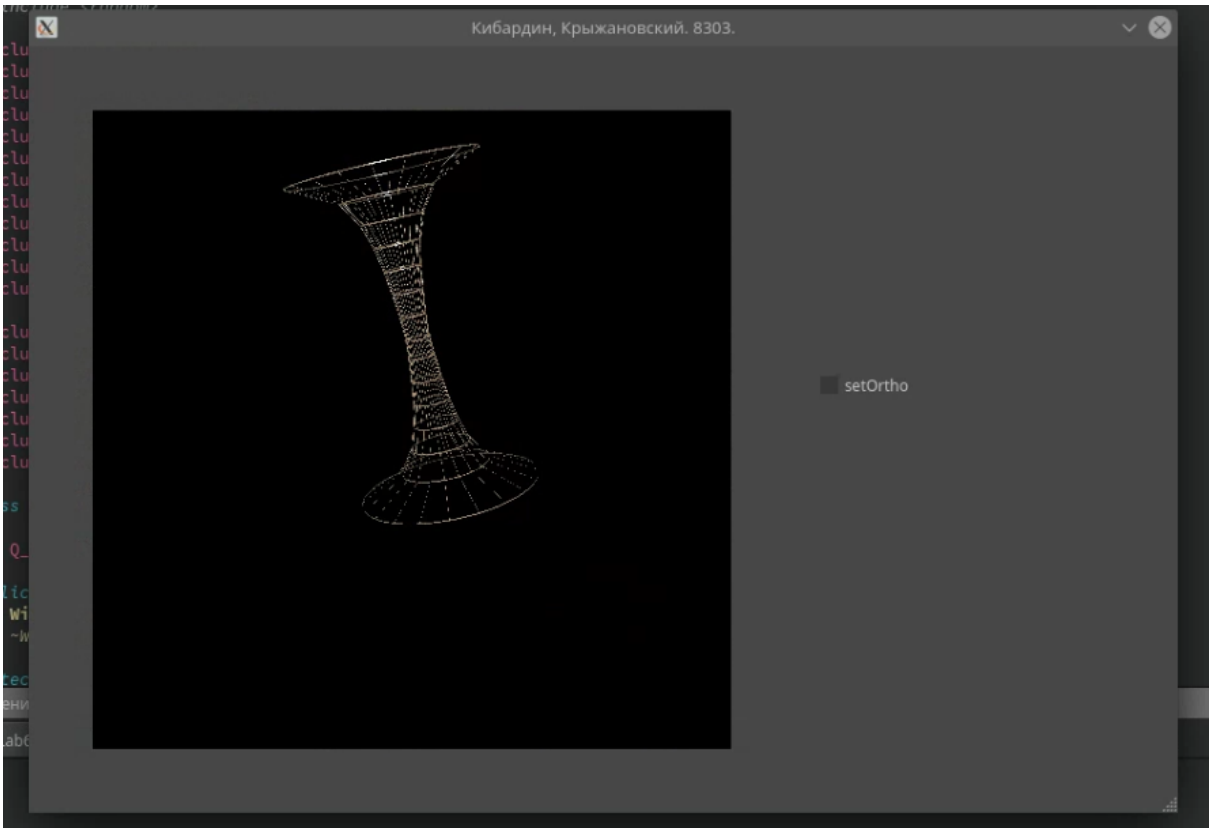
```
}
```

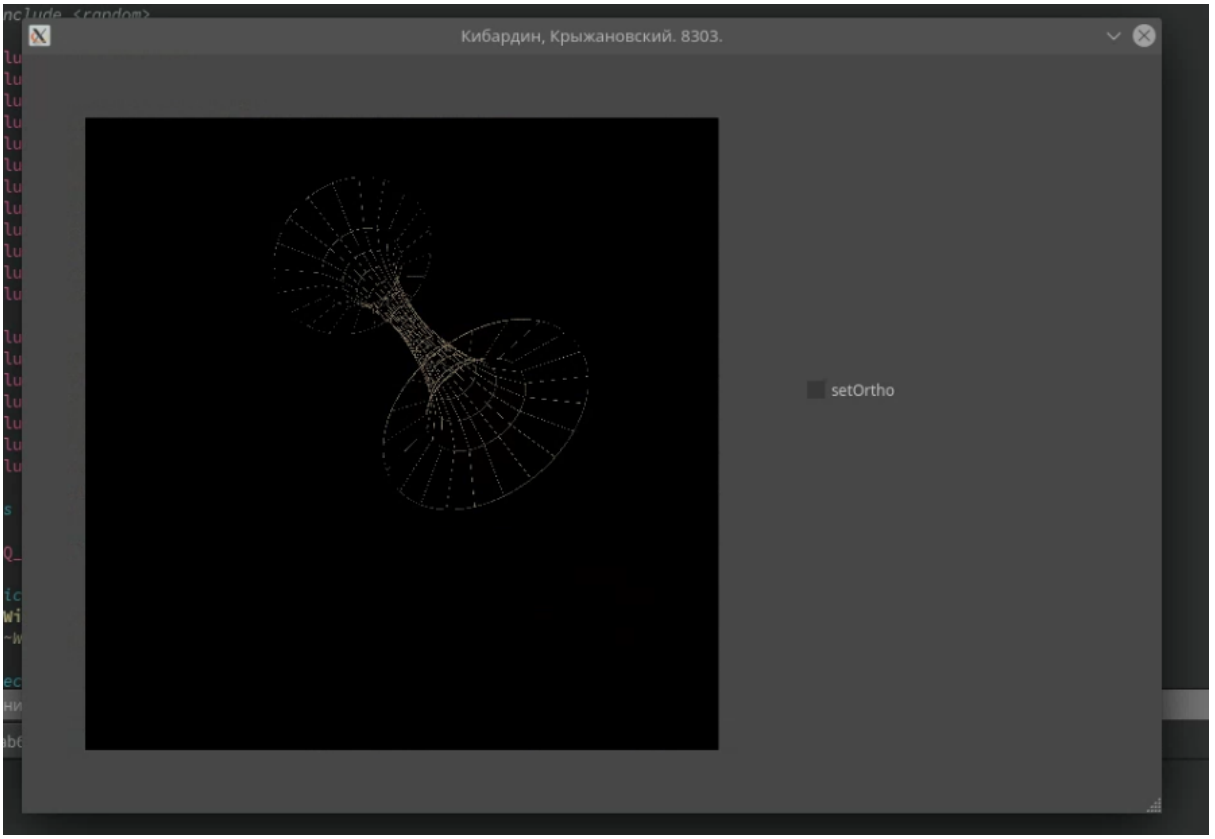
5. Код фрагментного шейдера

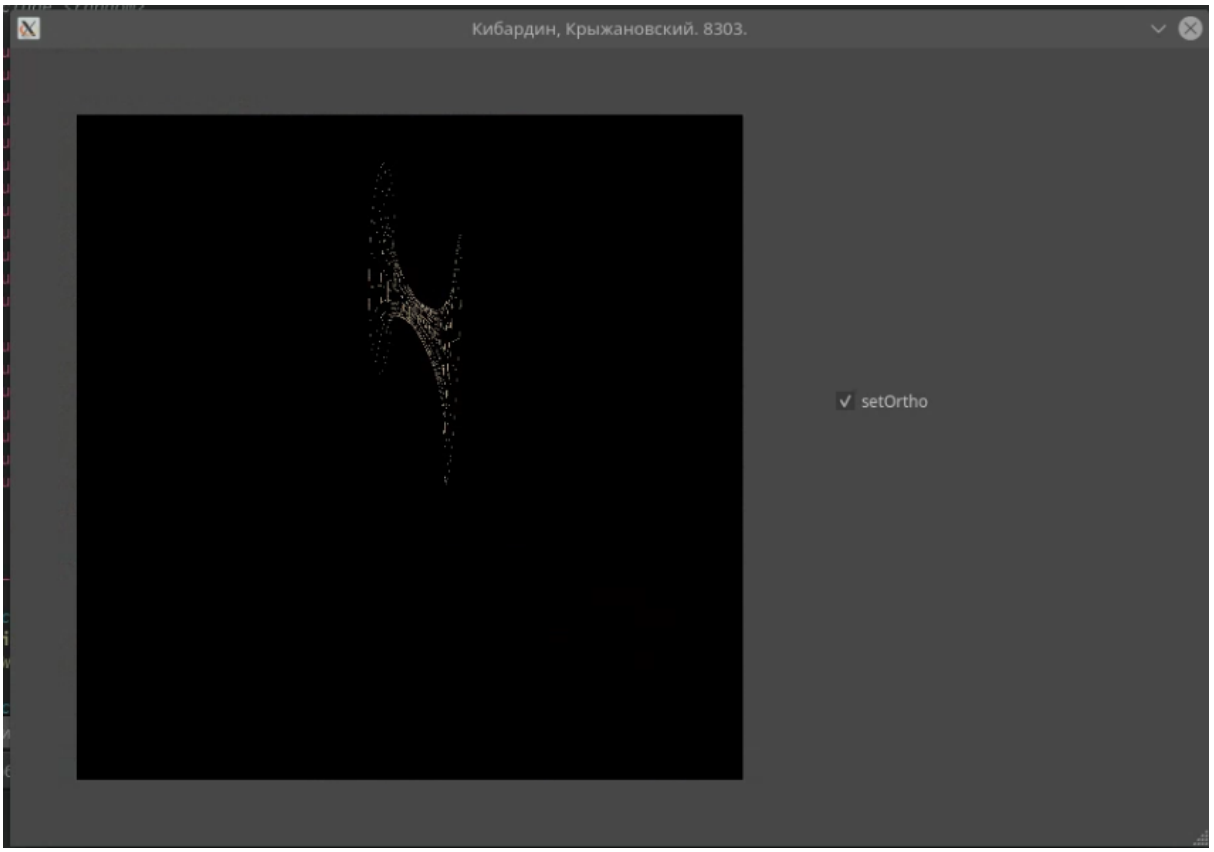
```
uniform sampler2D u_texture;
uniform highp vec4 u_lightPosition;
uniform highp float u_lightPower;

varying highp vec4 v_position;
varying highp vec2 v_texcoord;
varying highp vec3 v_normal;

void main(void)
{
   vec4 resultColor = vec4(0.0, 0.0, 0.0, 0.0);
   vec4 eyePosition = vec4(0.0, 0.0, 0.0, 1.0);
   vec4 diffuseMaterialColor = texture2D(u_texture, v_texcoord);
   vec3 eyeVector = normalize(v_position.xyz - eyePosition.xyz);
   vec3 lightVector = normalize(v_position.xyz - u_lightPosition.xyz);
   vec3 reflectLight = normalize(reflect(lightVector, v_normal));
   float len = length(v_position.xyz - eyePosition.xyz);
   float specularFactor = 50.0;
   float ambientFactor = 0.8;

      vec4 diffuseColor = diffuseMaterialColor * u_lightPower * max(0.0,
dot(v_normal, -lightVector));
   resultColor += diffuseColor;

   vec4 ambientColor = ambientFactor * diffuseMaterialColor;
   resultColor += ambientColor;

   vec4 specularColor = vec4(1.0, 1.0, 1.0, 1.0) * u_lightPower * pow(max(0.0,
dot(reflectLight, -eyeVector)), specularFactor);
   resultColor += specularColor;

   gl_FragColor = resultColor;
}
```

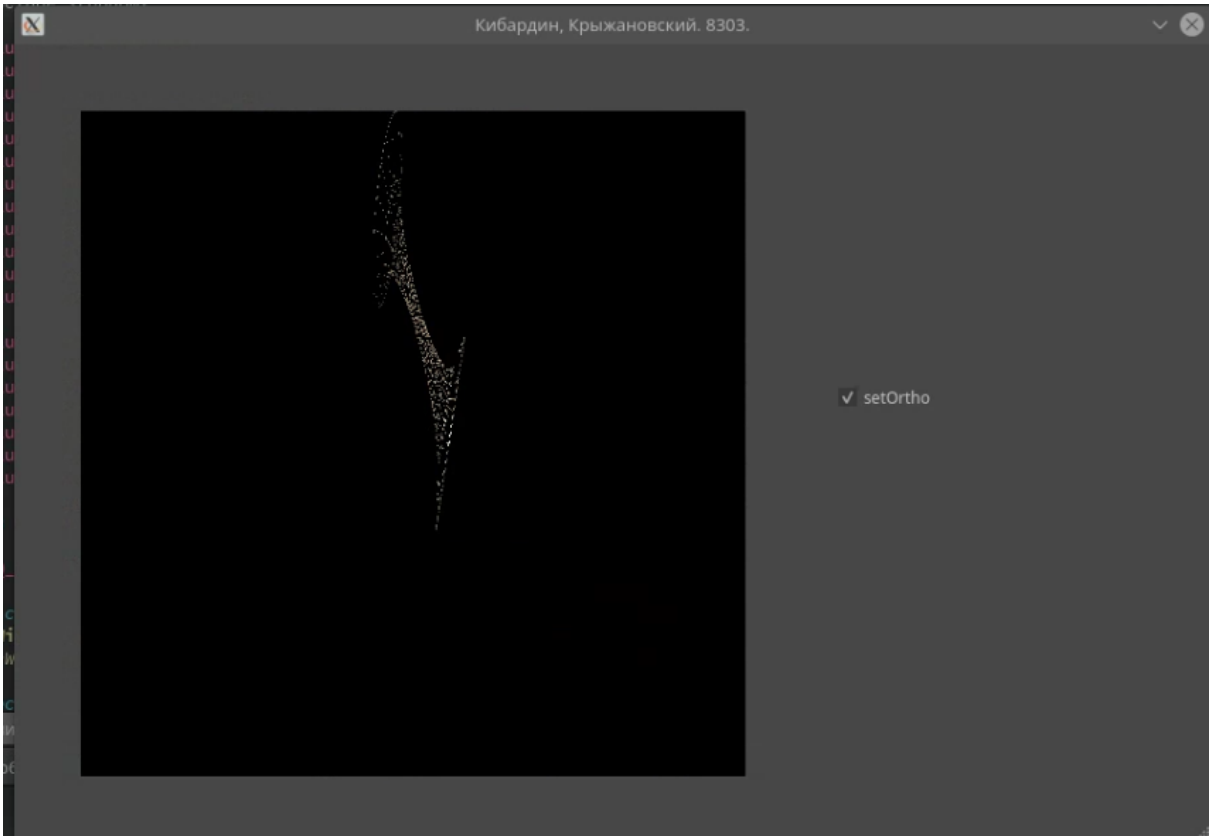
**Тестирование**

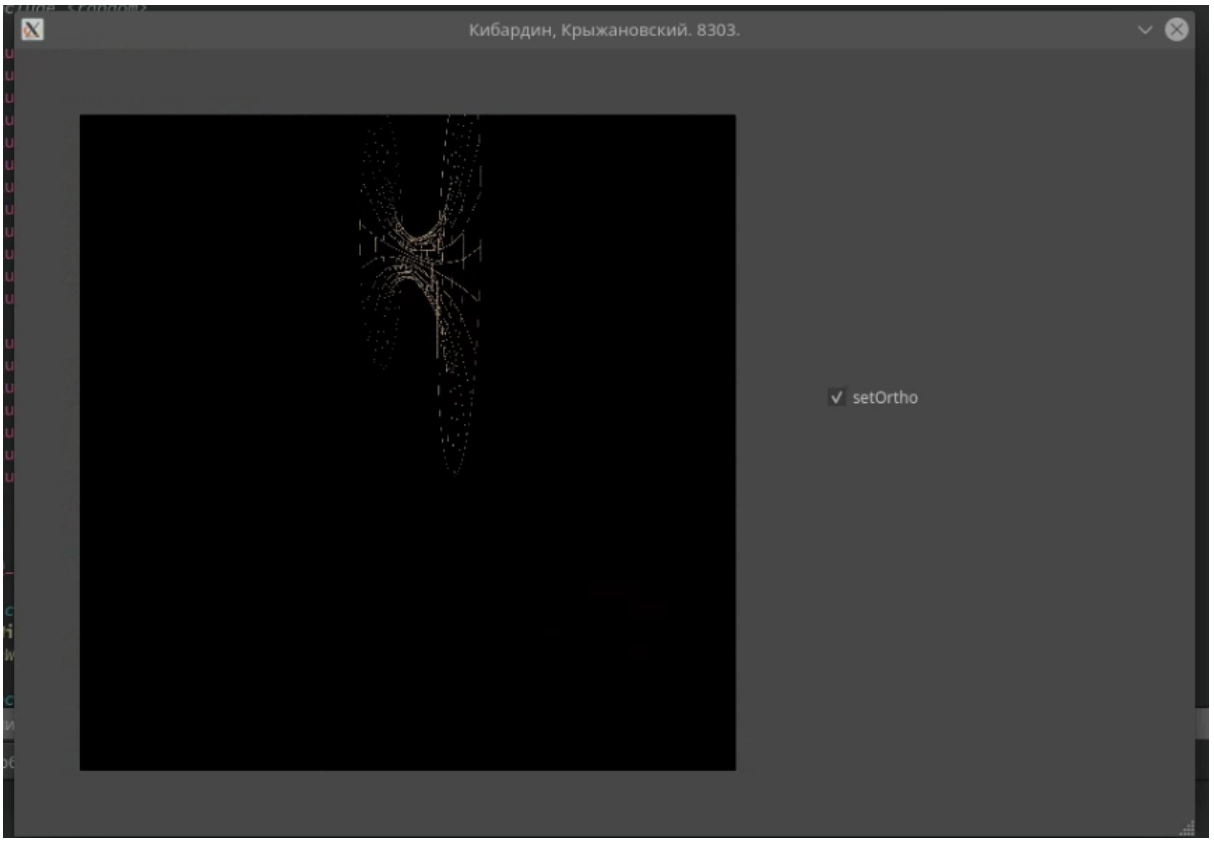
Результаты тестирования программы представлены на скриншотах ниже:

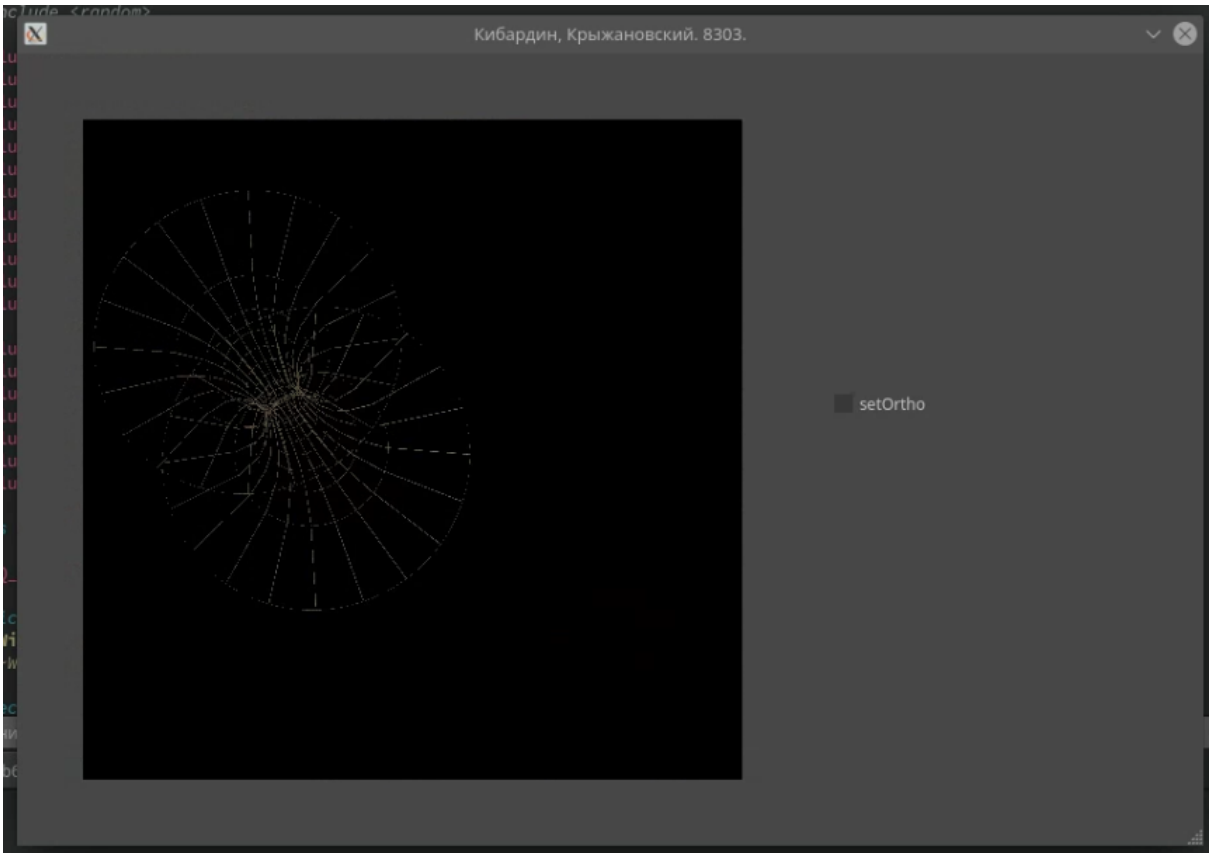



Кибардин, Крыжановский. 8303.
setOrtho


Кибардин, Крыжановский. 8303.
setOrtho


Кибардин, Крыжановский. 8303.
setOrtho


Кибардин, Крыжановский. 8303.
setOrtho

**Вывод**

В ходе выполнения лабораторной работы была написана программа, рисующая трехмерный объект с использованием шейдеров GLSL и OpenGL.