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

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Компьютерная графика»

Тема: Реализация трехмерного объекта с использованием библиотеки OpenGL

Студент гр.8382 _____ Нечепуренко Н.А.

Студент гр.8382 _____ Терехов А.Е.

Преподаватель _____ Герасимова Т.В.

Санкт-Петербург

2021

Цели работы.

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.

Задание.

Каркасные объекты

Написать программу, рисующую проекцию трехмерного каркасного объекта.

Требования

1. Грани объекта рисуются с помощью доступных функций рисования отрезка в координатах окна. При этом использовать шейдеры GLSL и OpenGL
2. Вывод многогранника с удалением или прорисовкой невидимых граней;
3. Ортогональное и перспективное проецирование;
4. Перемещения, повороты и масштабирование многогранника по каждой из осей независимо от остальных.
5. Генерация многогранника с заданной мелкостью разбиения.
6. Д.б. установлено изменение свойств источника света (интенсивность).
7. При запуске программы объект сразу должно быть хорошо виден.
8. Пользователь имеет возможность вращать фигуру (2 степени свободы) и изменять параметры фигуры.

9. Возможно изменять положение наблюдателя.
10. Нарисовать оси системы координат.
11. Все варианты требований могут быть выбраны интерактивно.

Задание 24 Усеченный эллипсоид

Выполнение работы.

Сгенерируем точки полигональной сетки эллипсоида в одном октанте с помощью библиотеки numpy.

```
def get_pts_in_octant(n_in_row: int, x, y, z) -> np.ndarray:
    a1 = x
    a2 = y
    a3 = z
    u = np.linspace(0, np.pi / 2, n_in_row)
    v = np.linspace(0, np.pi / 2, n_in_row)
    x = a1 * np.outer(np.cos(u), np.sin(v))
    y = a2 * np.outer(np.sin(u), np.sin(v))
    z = a3 * np.outer(np.ones(np.size(u)), np.cos(v))
    x = x.reshape(n_in_row ** 2)
    y = y.reshape(n_in_row ** 2)
    z = z.reshape(n_in_row ** 2)
    res = np.array(list(zip(x, y, z)))
    res = np.unique(res.round(decimals=5), axis=0)
    sort_ind = np.lexsort((res[:, 0], res[:, 1], -res[:, 2]))
    res = res[sort_ind]
    return res
```

Затем отразим это множество в оставшиемся 7 октантов.

```
def get_raw_pts(n_in_row: int, x, y, z) -> np.ndarray:
    assert n_in_row > 6
    result = get_pts_in_octant(n_in_row, x, y, z)
    result = np.append(result, result.dot(np.array([[-1, 0,
```

```
        0], [0, 1, 0], [0, 0, 1]])), 0)
    result = np.append(result, result.dot(np.array([[1, 0,
        0], [0, -1, 0], [0, 0, 1]])), 0)
    result = np.append(result, result.dot(np.array([[1, 0,
        0], [0, 1, 0], [0, 0, -1]])), 0)
    indices = get_indices(n_in_row)
    result = result[indices]
    return result
```

Для каждого треугольника сетки сгенерируем нормали для настройки освещения и отрисовки.

```
def add_normals(triangles: np.ndarray) -> np.ndarray:
    result = []
    for idx, triangle in enumerate(triangles):
        a = triangle[0]
        b = triangle[1]
        c = triangle[2]
        x = b - a
        y = c - a
        for _ in range(3):
            result.append(np.cross(x, y))
    result = normalize(np.array(result))
    triangles = triangles.reshape((triangles.shape[0] *
        triangles.shape[1], triangles.shape[2]))
    return np.append(triangles, result, 1)
```

Индексы вершин треугольников генерируются следующим образом

```
def get_indices(n_in_row: int) -> np.ndarray:
    indices = []
    for i in range(1, n_in_row):
        indices.append(0)
        indices.append(i)
        indices.append(i + 1)
```

```
    for i in range(1, n_in_row - 1):
        for j in range(1, n_in_row):
            indices.extend([(i - 1) * n_in_row + j, i *
               n_in_row + j, i * n_in_row + j + 1])
            indices.extend([(i - 1) * n_in_row + j, i *
               n_in_row + j + 1, (i - 1) * n_in_row + j + 1])
    indices = np.array(indices)
    octant_increment = indices.max()
    for i in range(3):
        indices = np.append(indices, np.flip(indices, 0) +
           octant_increment + 1)
        octant_increment = indices.max()
    return indices
```

В результате получается следующий объект (см. рис. 1).

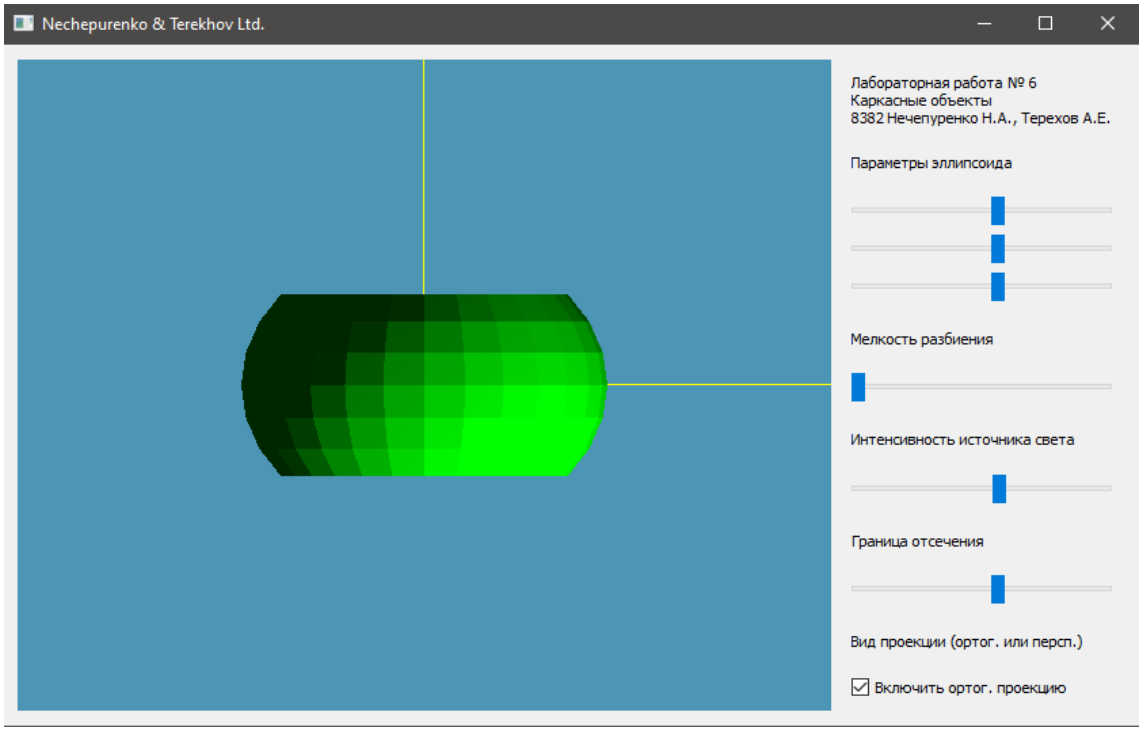

Рисунок 1 – Сгенерированный усеченный эллипсоид

Увеличим степень разбиения сетки (см. рис. 2).

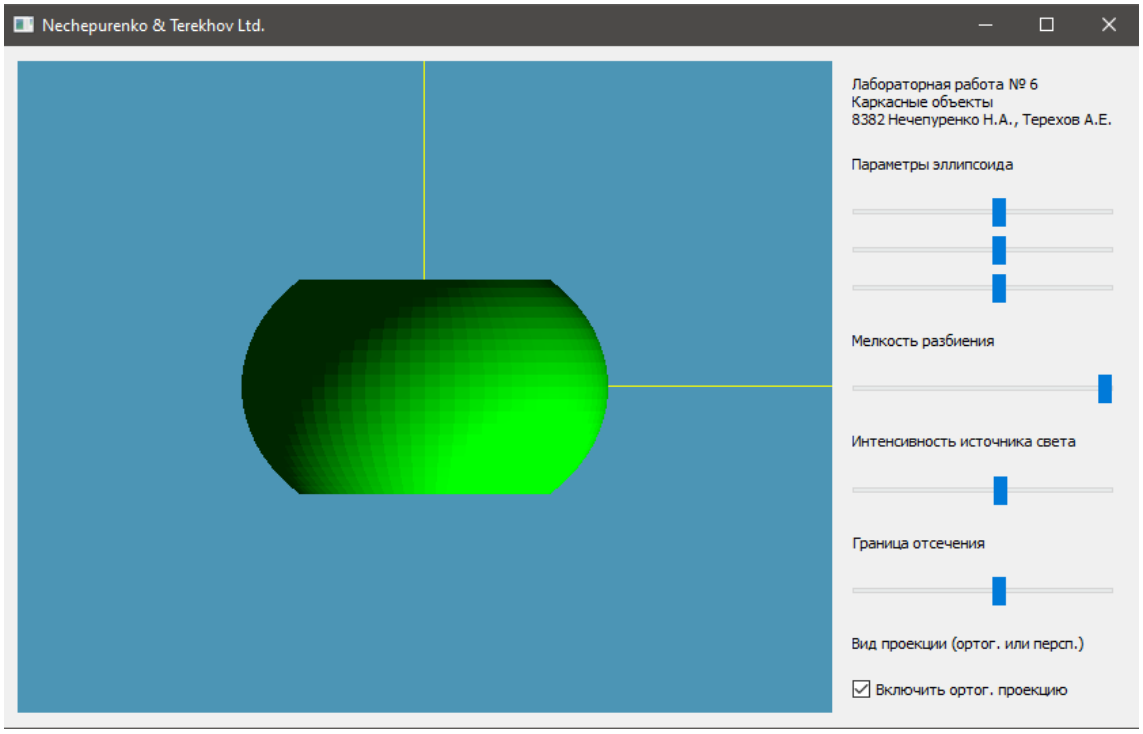

Рисунок 2 – Увеличенная степень разбиения

Сравним два разных вида проекции – ортогональную и перспективную. Для наглядности заглянем внутрь фигуры.

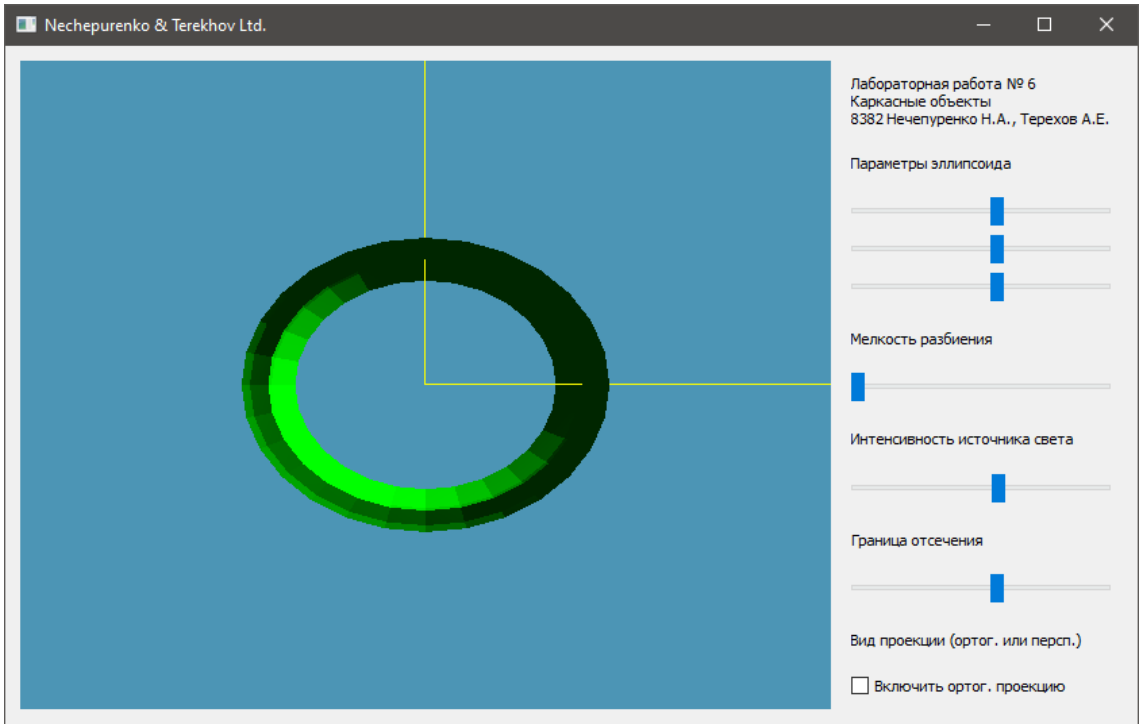

Рисунок 3 – Перспективная проекция

Видим внутреннюю часть эллипсоида, из-за схождения лучей в начало координат.

При ортогональной проекции такого эффекта не наблюдается.

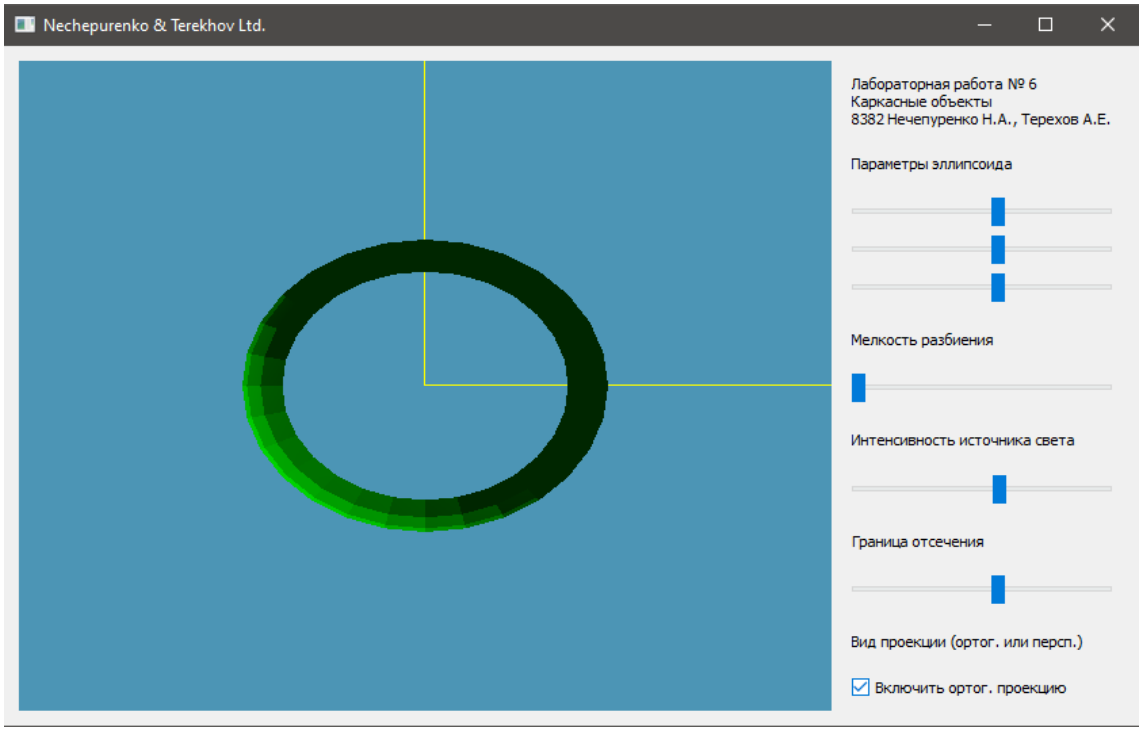

Рисунок 4 – Ортогональная проекция

Повращаем фигуры вокруг оси X и оси Y.

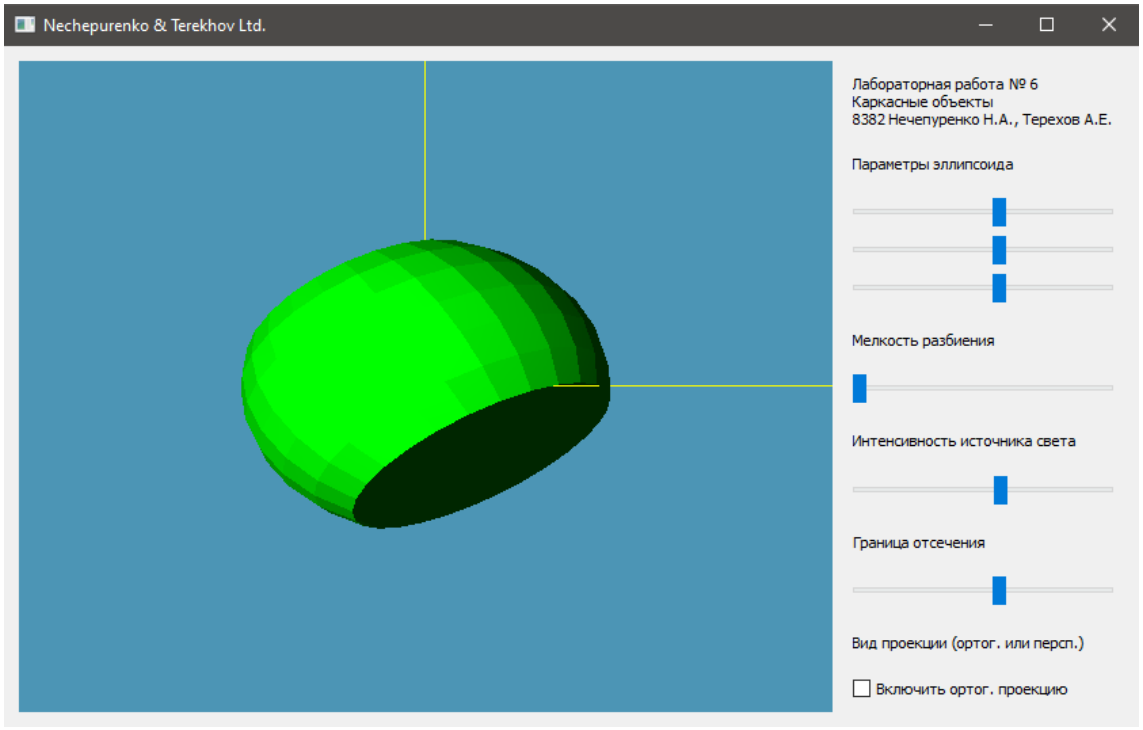

Рисунок 5 – Пример поворота по двум осям

Изменим длины осей эллипсоида.

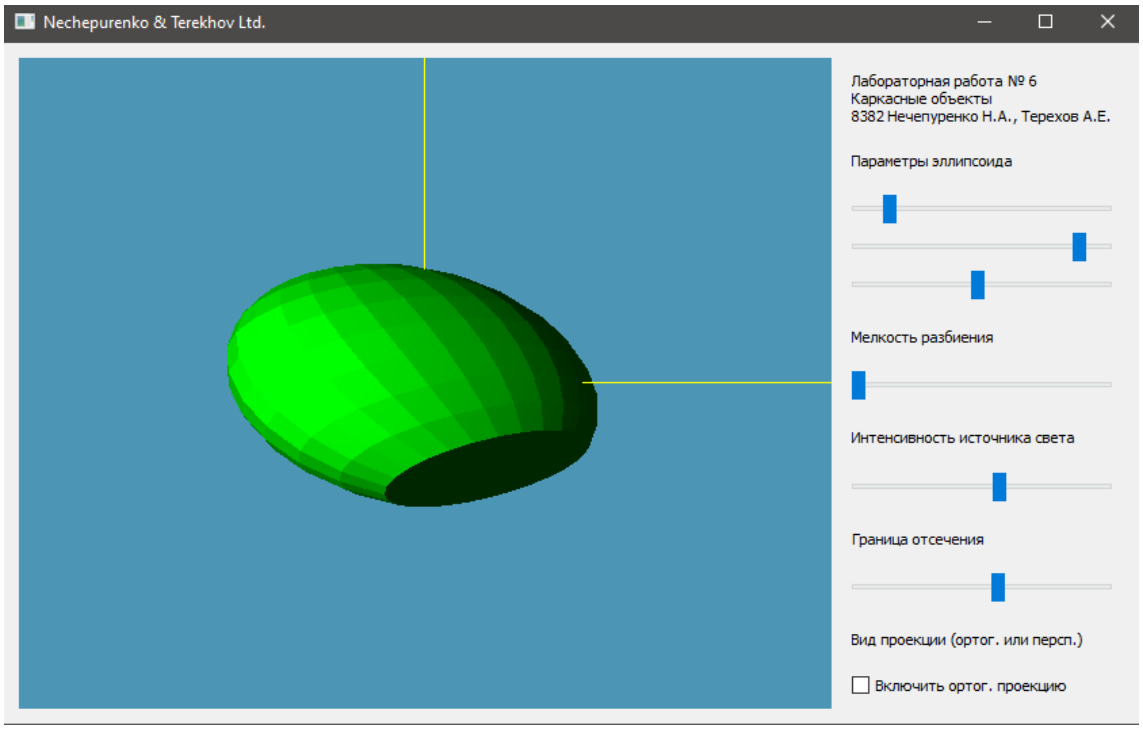

Рисунок 6 – Изменение длин осей

Увеличим границу отсечения.

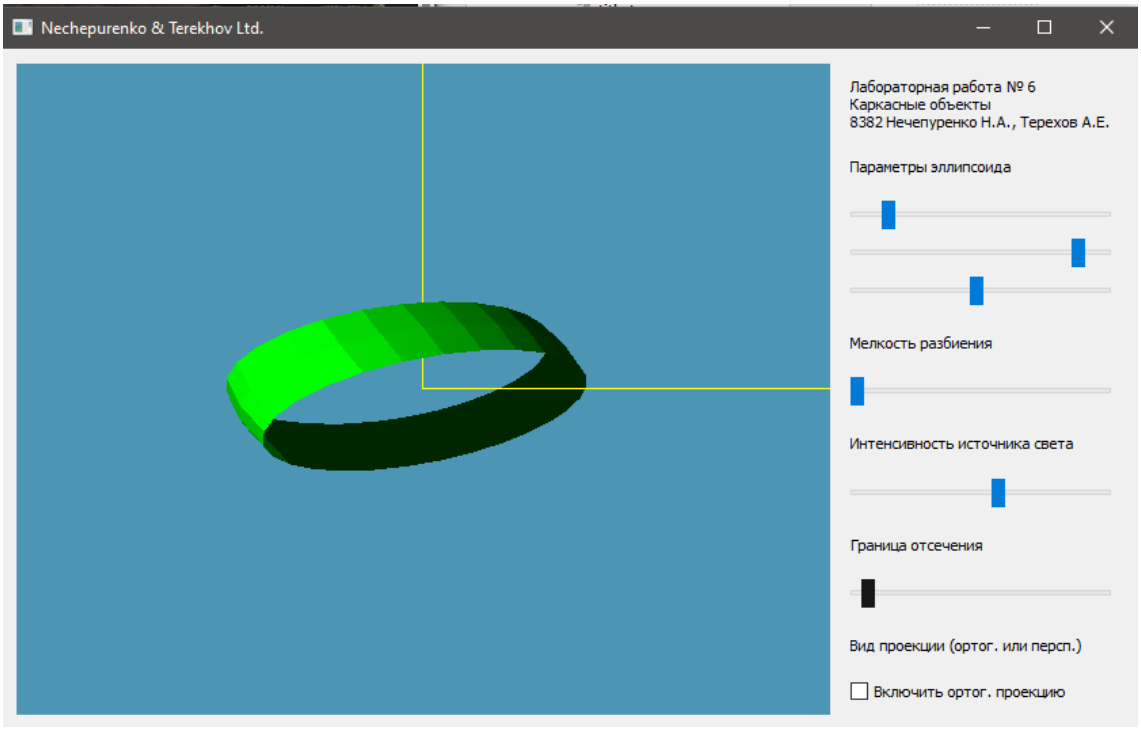

Рисунок 7 – Больший уровень отсечения

Изменим интенсивность источника света.

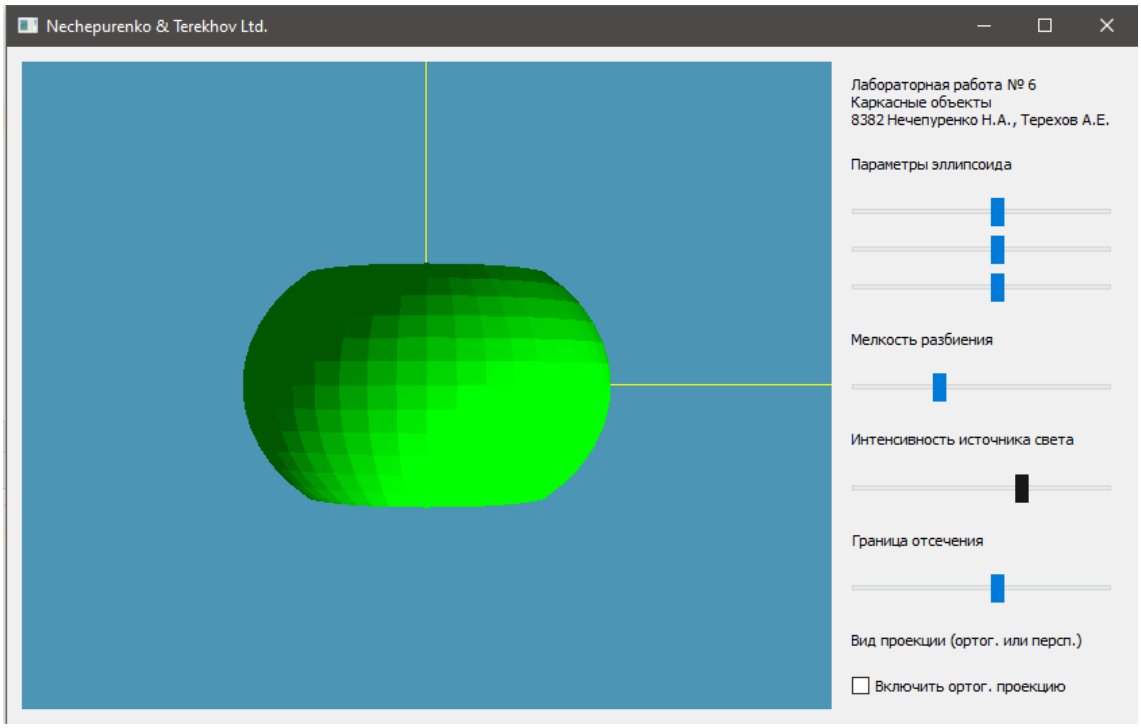

Рисунок 8 – Увеличенная интенсивность источника

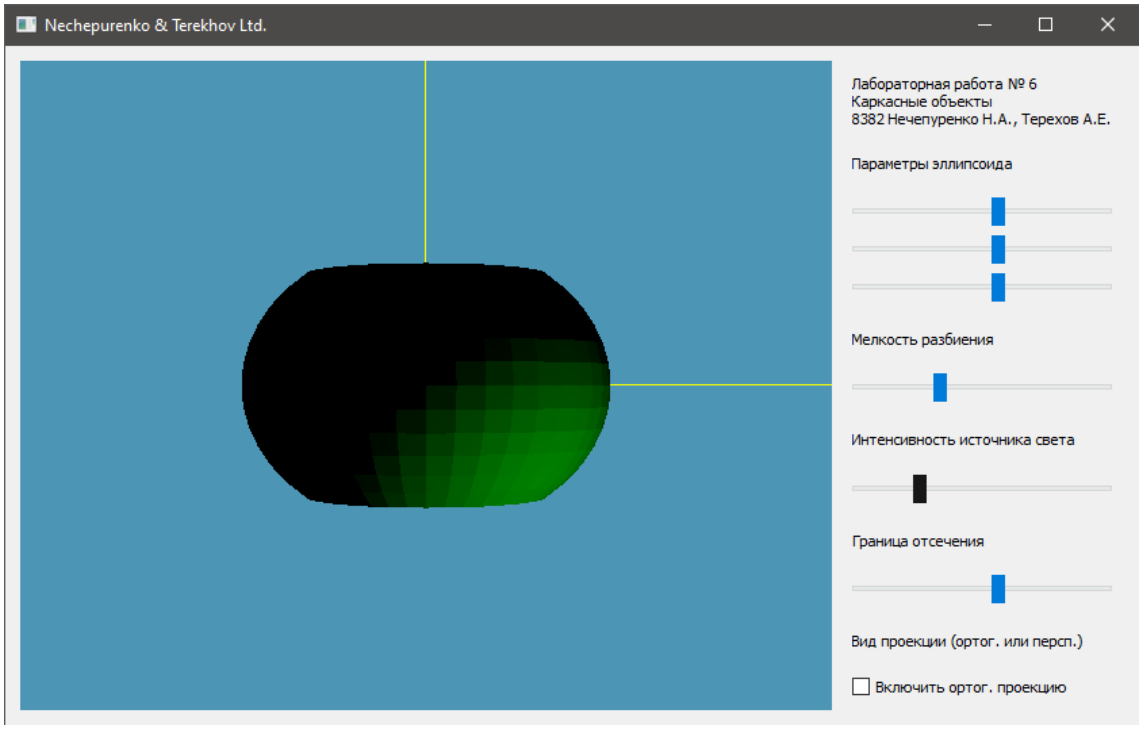

Рисунок 9 – Уменьшенная интенсивность источника

Все матричные преобразования производятся в шейдерах.

```
    #version 430

uniform mat4 projMat;
uniform mat4 viewMat;
uniform mat4 modelMat;
layout (location = 0) in vec3 position;
layout (location = 1) in vec3 normal;
out vec3 Normal;
out vec3 FragPos;

void main()
{
    FragPos = vec3(modelMat * vec4(vec3(position.x, position.z,
       position.y), 1.0));
    Normal = vec3(normal.x, normal.z, normal.y);
```

```
    gl_Position = projMat * viewMat * modelMat * vec4(vec3(
       position.x, position.z, position.y), 1.0);
}
```

Код фрагментного шейдера, в котором реализована модель освещения.

```
    #version 430

uniform vec3 lightPos;
uniform mat4 modelMat;
uniform float ambientStrength;
out vec4 fragColor;
in vec3 Normal;
in vec3 FragPos;

void main()
{
    vec3 lightColor = vec3(1.0, 1.0, 1.0);
    vec3 objectColor = vec3(0.0, 1.0, 0.0);
    vec3 ambient = ambientStrength * lightColor;
    vec4 lightPosTm = modelMat * vec4(lightPos, 1.0);
    vec3 lightPosTmp = vec3(lightPosTm.x, lightPosTm.y,
       lightPosTm.z);
    vec3 norm = normalize(Normal);
    vec3 lightDir = normalize(lightPosTmp - FragPos);
    float diff = max(dot(norm, lightDir), 0.0);
    vec3 diffuse = diff * lightColor;

    vec3 result = (ambient + diffuse) * objectColor;
    fragColor = vec4(result, 1.0);
}
```

Выводы.

В результате выполнения лабораторной работы была реализована программа, реализующую представление усеченного эллипсоида, используя предложенные функции библиотеки OpenGL (матрицы видового преобразования, проецирование) и язык GLSL. Разработанная программа обладает интерфейсом и обработчиком сигналов клавиатуры, что позволяет управлять различными параметрами генерации фигуры, ее отображения, вращения.