Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 3

по курсу «Компьютерная графика»

«Модельно-видовые и проективные преобразования»

Студент группы ИУ9-42Б Волохов А. В.

Преподаватель Цалкович П. А.

Москва 2024

1 Задача

Для выполнения лабораторной нужно будет взять код, который вы получили после лабораторной 2, отключить проективные преобразования и заменить куб на фигуру указанную в варианте.

Вариант: круговой тор

2 Основная теория

В данном коде реализован пример трехмерной графики с использованием библиотеки OpenGL и библиотеки GLFW для создания окна. Программа отображает тороидальную поверхность (тор) в пространстве.

Подход к инициализации окна и контекста OpenGL аналогичен предыдущему коду. Окно создается с использованием GLFW, а затем устанавливается текущий контекст OpenGL.

Функция display() отвечает за отрисовку сцены. Вначале происходит очистка буферов цвета и глубины. Затем устанавливается матрица проекции с помощью функции projection(). В данном случае происходит вращение сцены вокруг оси X и Y. После этого вызывается функция torus(), которая отрисовывает тор, используя примитивы OpenGL.

Функции key callback() и scroll callback() отвечают за обработку нажатий клавиш и колеса мыши соответственно. При нажатии клавиш стрелок происходит изменение углов поворота сцены. При прокрутке колеса мыши изменяется размер тора.

3 Практическая реализация

Код представлен в Листинге 1.

Листинг 1 - lab3.py

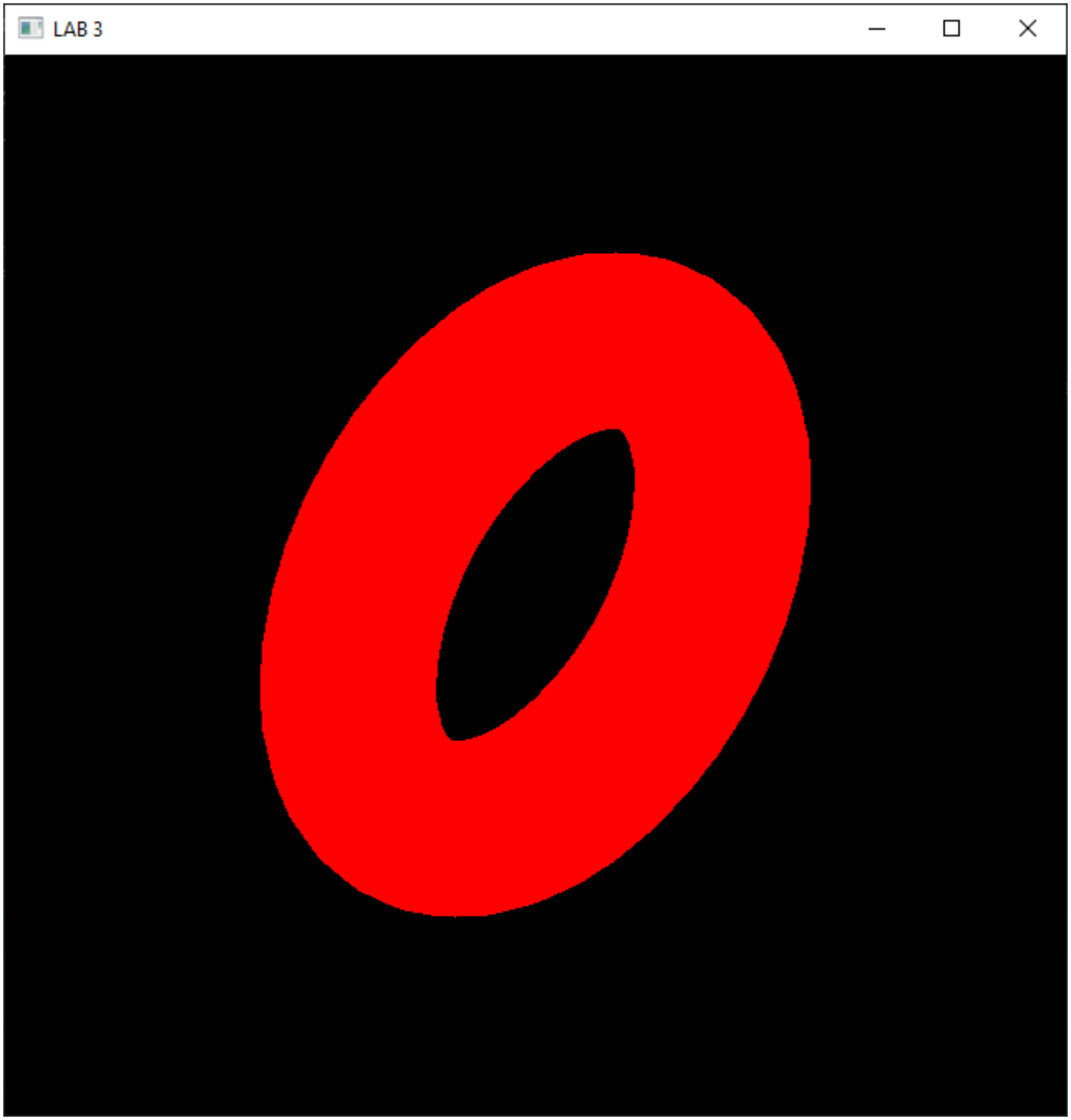
```
import math

import glfw
import numpy as np
from OpenGL.GL import *
from math import cos, sin

alpha = 0
beta = 0
size = 0.5
fill = True


def main():
    if not glfw.init():
        return
    window = glfw.create_window(640, 640, "LAB 3", None, None)
    if not window:
        glfw.terminate()
        return
    glfw.make_context_current(window)
    glfw.set_key_callback(window, key_callback)
    glfw.set_scroll_callback(window, scroll_callback)

    glEnable(GL_DEPTH_TEST)
    glDepthFunc(GL_LESS)
    while not glfw.window_should_close(window):
        display(window)
    glfw.destroy_window(window)
    glfw.terminate()


def display(window):
    glLoadIdentity()
    glClear(GL_COLOR_BUFFER_BIT)
    glClear(GL_DEPTH_BUFFER_BIT)
    glMatrixMode(GL_PROJECTION)
    global alpha
    global beta
```

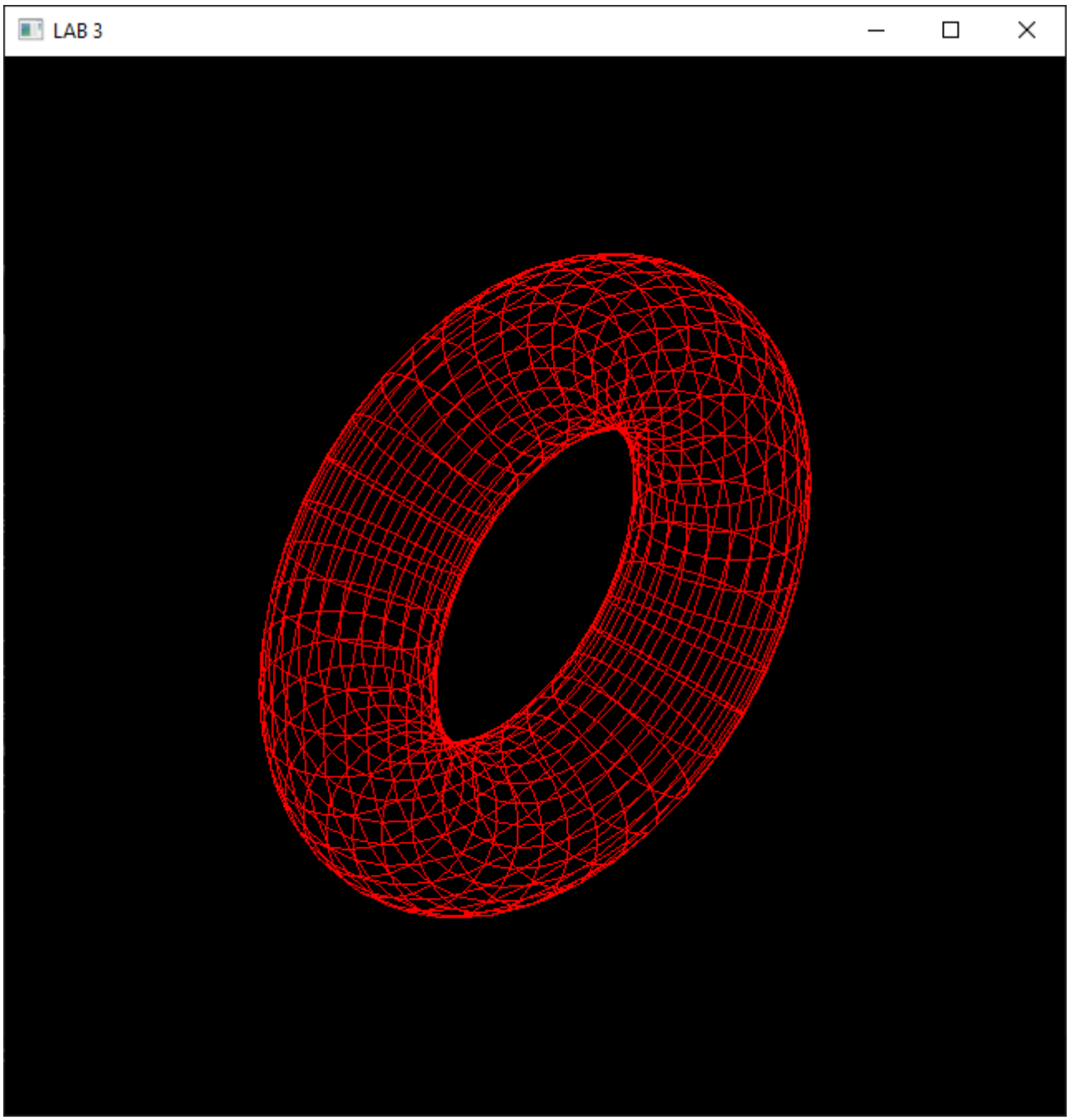
```
    def projection():
        alpha_rad = np.radians(alpha)
        beta_rad = np.radians(beta)

        rotate_y = np.array([
            [cos(alpha_rad), 0, sin(alpha_rad), 0],
            [0, 1, 0, 0],
            [-sin(alpha_rad), 0, cos(alpha_rad), 0],
            [0, 0, 0, 1]
        ])

        rotate_x = np.array([
            [1, 0, 0, 0],
            [0, cos(beta_rad), -sin(beta_rad), 0],
            [0, sin(beta_rad), cos(beta_rad), 0],
            [0, 0, 0, 1]
        ])

        glMultMatrixf(rotate_x)
        glMultMatrixf(rotate_y)

    def torus(R, r, N, n):
        for i in range(N):
            for j in range(n):
                theta = (2 * math.pi / N) * i
                phi = (2 * math.pi / n) * j
                theta_next = (2 * math.pi / N) * (i + 1)
                phi_next = (2 * math.pi / n) * (j + 1)

                #               "                "

                x0 = (R + r * cos(phi)) * cos(theta)
                y0 = (R + r * cos(phi)) * sin(theta)
                z0 = r * sin(phi)

                x1 = (R + r * cos(phi)) * cos(theta_next)
                y1 = (R + r * cos(phi)) * sin(theta_next)
                z1 = r * sin(phi)

                x2 = (R + r * cos(phi_next)) * cos(theta_next)
                y2 = (R + r * cos(phi_next)) * sin(theta_next)
                z2 = r * sin(phi_next)

                x3 = (R + r * cos(phi_next)) * cos(theta)
                y3 = (R + r * cos(phi_next)) * sin(theta)
                z3 = r * sin(phi_next)
```

```
                glBegin(GL_QUADS)
                glVertex3f(x0, y0, z0)
                glVertex3f(x1, y1, z1)
                glVertex3f(x2, y2, z2)
                glVertex3f(x3, y3, z3)
                glEnd()

    glLoadIdentity()

    projection()
    R = size
    r = size / 3

    glColor3f(1.0, 0.0, 0.0)

    torus(R, r, 40, 25)

    glfw.swap_buffers(window)
    glfw.poll_events()


def key_callback(window, key, scancode, action, mods):
    global alpha
    global beta
    if action == glfw.PRESS or action == glfw.REPEAT:
        if key == glfw.KEY_RIGHT:
            alpha += 3
        elif key == glfw.KEY_LEFT:
            alpha -= 3
        elif key == glfw.KEY_UP:
            beta -= 3
        elif key == glfw.KEY_DOWN:
            beta += 3
        elif key == glfw.KEY_F:
            global fill
            fill = not fill
            if fill:
                glPolygonMode(GL_FRONT_AND_BACK, GL_FILL)
            else:
                glPolygonMode(GL_FRONT_AND_BACK, GL_LINE)


def scroll_callback(window, xoffset, yoffset):
    global size
```

```
    if xoffset > 0:
        size -= yoffset / 10
    else:
        size += yoffset / 10


if __name__ == "__main__":
    main()
```

В результате работы программы получился следующий вывод:




LAB 3

4 Заключение

В результате выполнения лабораторной работы был создан пример трехмерной графики с помощью библиотеки OpenGL. Реализована отрисовка тороидальной поверхности, а также добавлена возможность вращения и изменения размера объекта с помощью клавиш и колеса мыши. Этот пример демонстрирует основные принципы работы с трехмерной графикой в контексте OpenGL и может быть использован в дальнейшем для изучения более сложных трехмерных моделей и алгоритмов отображения.