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 No 2

по курсу «Компьютерная графика»

Студент группы ИУ9-41Б Пишикина М. В.

Преподаватель Цалкович П. А.

Москва 2024

1 Задача

1. Определить круговой тор в качестве модели объекта сцены.

2. Добавить в сцену круговой тор (в стандартной ориентации, не изменяемой при модельно-видовых преобразованиях основного объекта).

3. Реализовать изменение ориентации и размеров объекта (навигацию камеры) с помощью модельно-видовых преобразований (без gluLookAt). Управление производится интерактивно с помощью клавиатуры и/или мыши.

4. Предусмотреть возможность переключения между каркасным и твердотельным отображением модели (glFrontFace/ glPolygonMode).

Вид проекции - горизонтальная изометрия

2 Основная теория

- glPushMatrix() #сохраняет текущую матрицу моделирования
- glPopMatrix() # восстанавливает текущую матрицу моделирования
- glGetIntegerv (GL_MAX_MODELVIEW_STACK_DEPTH) #определение глуб
- glGetIntegerv (GL_MODELVIEW_STACK_DEPTH) #определение текущей гл
- Преобразования объектов и камеры в OpenGL производятся с помощью умножения векторов координат на текущую матрицу в момент определения координат вершин

3 Практическая реализация

```
import glfw
import math
from math import cos, sin, sqrt, asin
from OpenGL.GL import *

WINDOW_HEIGHT = 600
WINDOW_WIDTH = 600

angleX = 0.0
```

```
angleZ= 0.0
delta = math.pi / 16
size = 1
fill = False

frontViewMatrix = [1, 0, 0, 0,
                   0, 1, 0, 0,
                   0, 0, -1, 0,
                   0, 0, 0, 1]
sideViewMatrix = [0, 0, -1, 0,
                  0, 1, 0, 0,
                  -1, 0, 0, 0,
                  0, 0, 0, 1]
topViewMatrix = [1, 0, 0, 0,
                 0, 0, -1, 0,
                 0, -1, 0, 0,
                 0, 0, 0, 1]


def main():
    if not glfw.init():
        return
    window = glfw.create_window(WINDOW_WIDTH, WINDOW_HEIGHT, "л
    if not window:
        glfw.terminate()
        return
    glfw.make_context_current(window)
    glEnable(GL_DEPTH_TEST)
    glfw.set_key_callback(window, key_callback)
    glfw.set_scroll_callback(window, scroll_callback)
    glfw.set_mouse_button_callback(window, mouse_button_callback)
    while not glfw.window_should_close(window):
        display(window)
```

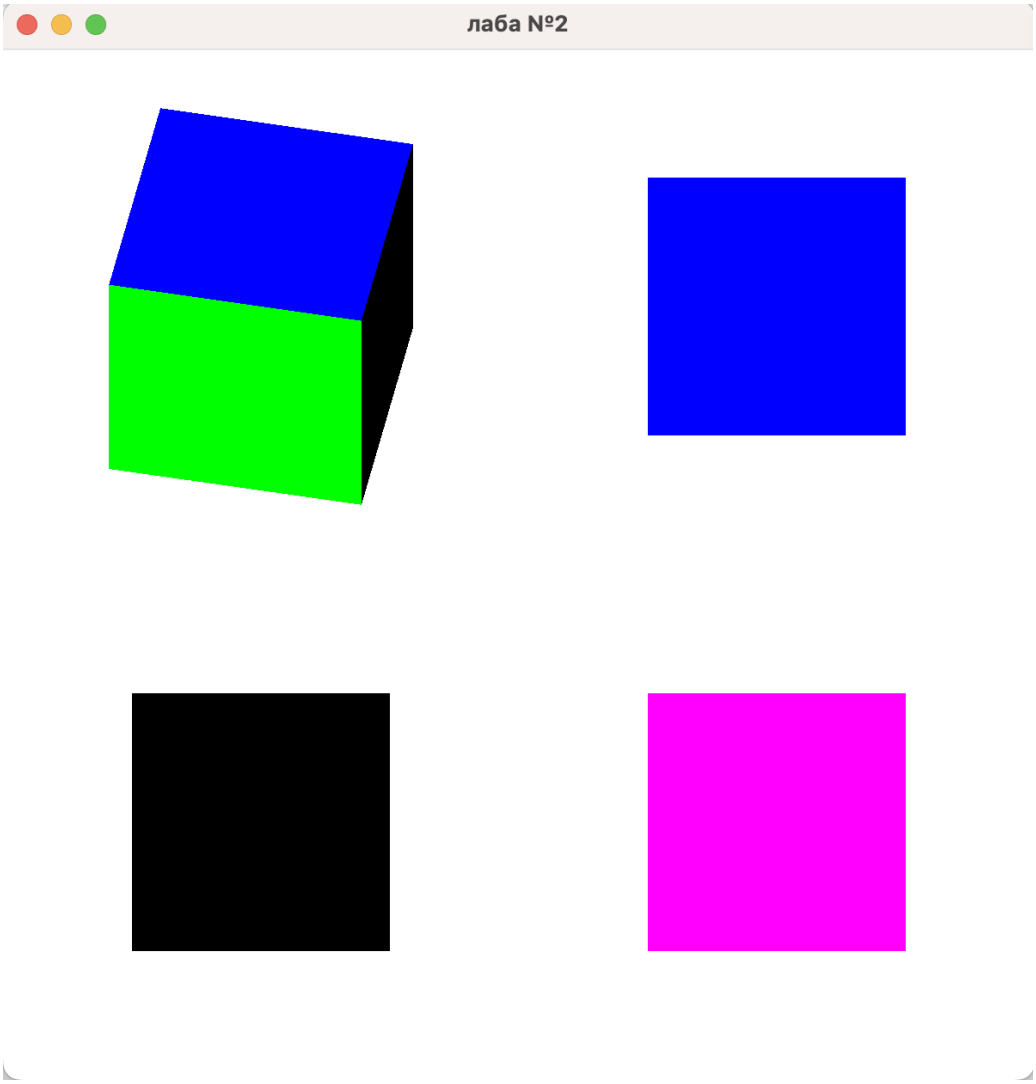
```
    glfw.destroy_window(window)
    glfw.terminate()

def display(window):
    theta = 0.775
    phi = 1.37

    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glClearColor(1.0, 1.0, 1.0, 1.0)

    #glRotatef(50, 1, 0, 1)

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

    glViewport(0, WINDOW_HEIGHT, WINDOW_WIDTH, WINDOW_HEIGH
    glMatrixMode(GL_PROJECTION)
    glLoadMatrixf([0.19945, 0.685, 0.499, 0, 0, 0.714, -0.67, 0, 0.979, -0.1395, -0.142
    draw_cube()

    glMatrixMode(GL_MODELVIEW) # крутим относительно Z
    glLoadMatrixf([math.cos(angleZ), math.sin(angleZ), 0, 0, -math.sin(angleZ), ma
    # умножаем на кручение относительно X
    glMultMatrixf([1, 0, 0, 0, 0, math.cos(angleX), -math.sin(angleX), 0, 0, math.si
    # умножаем на матрицу изменения размера
    glMultMatrixf([size, 0, 0, 0, 0, size, 0, 0, 0, 0, size, 0, 0, 0, 0, 1])

    glViewport(WINDOW_WIDTH, WINDOW_HEIGHT, WINDOW_WIDTH,
    glMatrixMode(GL_PROJECTION)
    glLoadMatrixf(topViewMatrix)
    draw_cube()

    glViewport(0, 0, WINDOW_WIDTH, WINDOW_HEIGHT)
```

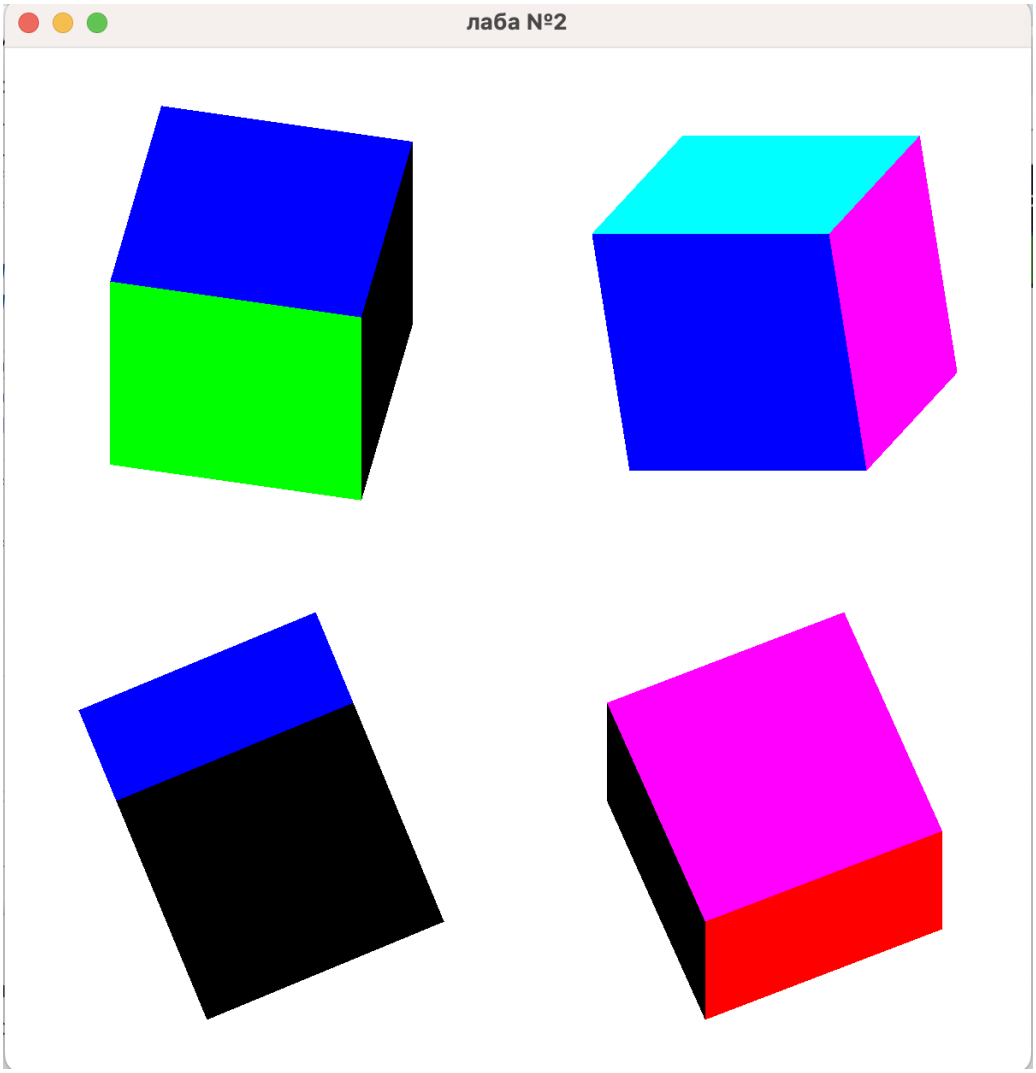
```
        glMatrixMode(GL_PROJECTION)
        glLoadMatrixf(frontViewMatrix)
        draw_cube()

        glViewport(WINDOW_WIDTH, 0, WINDOW_WIDTH, WINDOW_HEIGHT
        glMatrixMode(GL_PROJECTION)
        glLoadMatrixf(sideViewMatrix)
        draw_cube()

        glfw.swap_buffers(window)
        glfw.poll_events()

def draw_cube():
        glBegin(GL_POLYGON)
        glColor3f( 0.0, 1.0, 1.0 )
        glVertex3f(  0.5, -0.5, -0.5 )
        glVertex3f(  0.5,  0.5, -0.5 )
        glVertex3f( -0.5,  0.5, -0.5 )
        glVertex3f( -0.5, -0.5, -0.5 )
        glEnd()

        glBegin(GL_POLYGON)
        glColor3f(   0.0,  0.0, 0.0 )
        glVertex3f(  0.5, -0.5, 0.5 )
        glVertex3f(  0.5,  0.5, 0.5 )
        glVertex3f( -0.5,  0.5, 0.5 )
        glVertex3f( -0.5, -0.5, 0.5 )
        glEnd()

        glBegin(GL_POLYGON)
        glColor3f(  1.0,  0.0,  1.0 )
        glVertex3f( 0.5, -0.5, -0.5 )
        glVertex3f( 0.5,  0.5, -0.5 )
```

```
        glVertex3f( 0.5,  0.5,  0.5 )
        glVertex3f( 0.5, -0.5,  0.5 )
        glEnd()

        glBegin(GL_POLYGON)
        glColor3f(   0.0,  1.0,  0.0 )
        glVertex3f( -0.5, -0.5,  0.5 )
        glVertex3f( -0.5,  0.5,  0.5 )
        glVertex3f( -0.5,  0.5, -0.5 )
        glVertex3f( -0.5, -0.5, -0.5 )
        glEnd()

        glBegin(GL_POLYGON)
        glColor3f(   0.0,  0.0,  1.0 )
        glVertex3f(  0.5,  0.5,  0.5 )
        glVertex3f(  0.5,  0.5, -0.5 )
        glVertex3f( -0.5,  0.5, -0.5 )
        glVertex3f( -0.5,  0.5,  0.5 )
        glEnd()

        glBegin(GL_POLYGON)
        glColor3f(   1.0,  0.0,  0.0 )
        glVertex3f(  0.5, -0.5, -0.5 )
        glVertex3f(  0.5, -0.5,  0.5 )
        glVertex3f( -0.5, -0.5,  0.5 )
        glVertex3f( -0.5, -0.5, -0.5 )
        glEnd()

def key_callback(window, key, scancode, action, mods):
        global angleX, angleZ, size, fill
        if action == glfw.PRESS:
                if key == glfw.KEY_UP:
                        angleX += delta
```

```
            elif key == glfw.KEY_DOWN:
                angleX -= delta
            elif key == glfw.KEY_RIGHT:
                angleZ += delta
            elif key == glfw.KEY_LEFT:
                angleZ -= delta
            elif key == glfw.KEY_A:
                size += 0.1
            elif key == glfw.KEY_D:
                size -= 0.1
            elif key == glfw.KEY_SPACE:
                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

def mouse_button_callback(window, button, action, mods):
    global size
    if button == glfw.MOUSE_BUTTON_RIGHT and action == glfw.PRESS:
        size -= 0.1
    elif button == glfw.MOUSE_BUTTON_LEFT and action == glfw.PRESS:
        size += 0.1

main()
```

4 Пример работы программы (отправка на свою почту)





5 Заключение

В моей программе использовалась библиотека OpenGL, в которой реализованы основные функции для отрисовки фигур и их анимации. Мой вариант задания требовал реализации кругового сектора. Эта фигура должна была быть анимирована, а также должна была присутствовать неподвижная версия этой же фигуры. С помощью заданных матриц и соответствующих функций я успешно реализовала функционал анимации кругового сектора, его масштабирования, изменения цвета и заливки.