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 № 2

по курсу «Алгоритмы компьютерной графики»

Студент группы ИУ9-41Б Утебаева М. Б.

Преподаватель Цалкович П. А.

Москва 2024

1 Задача

1. Определить куб в качестве модели объекта сцены.

2. Определить преобразования, позволяющие получить фронтальную диметрию, добавить в сцену куб (в стандартной ориентации, не изменяемой при модельно-видовых преобразованиях основного объекта) для демонстрации проекции.

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).

2 Основная теория

Геометрические преобразования представляются матрицами, с помощью них можно осуществлять перенос, масштабирование, сдвиг и поворот объекта.

Есть три типа матриц преобразования, над которыми можно выполнять операции: 1) видовая матрица – определяет преобразования объекта в мировых координатах, такие как параллельный перенос, изменение масштаба и поворот; 2) матрица проекций – определяет, как будут проецироваться трехмерные объекты на плоскость экрана (в оконные координаты); 3) матрица текстуры определяет наложение текстуры на объект.

```
glMatrixMode(GL_PROJECTION) #включение режима работы с матрицей проекций
glMatrixMode(GL_MODELVIEW) #включение режима работы с модельно-видовой матрицей
glLoadIdentity() #замена текущей матрицы на единичную
glMultMatrixf(T) #умножение текущей матрицы на матрицу T
glEnable(GL_DEPTH_TEST) #отсечение заслонённых фигур
glDepthFunc(GL_LESS) #обработка глубины
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL) #перевод объекта в твёрдотельный
glPolygonMode(GL_FRONT_AND_BACK, GL_LINE) #перевод объекта в каркасный
glBegin(GL_QUADS) #задание прямоугольников из 4-х вершин
```

3 Практическая реализация

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

alpha = 0.0
beta = 0.0
fill = False

def cube(sz):
    glBegin(GL_QUADS)
    #левая
    glColor3f(0.1, 0.0, 0.7)
    glVertex3f(-sz, -sz, -sz)
    glVertex3f(-sz, sz, -sz)
    glVertex3f(-sz, sz, sz)
    glVertex3f(-sz, -sz, sz)
```

```
    #правая
    glColor3f(0.4, 0.9, 0.0)
    glVertex3f(sz, -sz, -sz)
    glVertex3f(sz, -sz, sz)
    glVertex3f(sz, sz, sz)
    glVertex3f(sz, sz, -sz)
    #нижняя
    glColor3f(0.0, 1.0, 1.0)
    glVertex3f(-sz, -sz, -sz)
    glVertex3f(-sz, -sz, sz)
    glVertex3f(sz, -sz, sz)
    glVertex3f(sz, -sz, -sz)
    #верхняя
    glColor3f(0.0, 0.5, 0.8)
    glVertex3f(-sz, sz, -sz)
    glVertex3f(-sz, sz, sz)
    glVertex3f(sz, sz, sz)
    glVertex3f(sz, sz, -sz)
    #передняя
    glColor3f(0.5, 0.0, 0.8)
    glVertex3f(-sz, -sz, -sz)
    glVertex3f(sz, -sz, -sz)
    glVertex3f(sz, sz, -sz)
    glVertex3f(-sz, sz, -sz)
    #задняя
    glColor3f(1.0, 1.0, 1.0)
    glVertex3f(-sz, -sz, sz)
    glVertex3f(sz, -sz, sz)
    glVertex3f(sz, sz, sz)
    glVertex3f(-sz, sz, sz)

    glEnd()

def display(window):
    glEnable(GL_DEPTH_TEST) #отсечение заслонённых фигур
    glDepthFunc(GL_LESS) #обработка глубины(?)
    glClear(GL_COLOR_BUFFER_BIT)
    glClear(GL_DEPTH_BUFFER_BIT)

    glMatrixMode(GL_PROJECTION) #задаём видимое пространство
    glLoadIdentity() #единичная матрица
    T = [1, 0, 0, 0,
         0, 1, 0, 0,
         0.5*cos(pi/4), 0.5*sin(pi/4), 1, 0,
         0, 0, 0, 1] #с какого ракурса видим объект
    glMultMatrixf(T)
```

```
    glMatrixMode(GL_MODELVIEW) #матрица представления модели
    glLoadIdentity()
    Z = [cos(beta), 0, -sin(beta), 0,
         0, 1, 0, 0,
         sin(beta), 0, cos(beta), 0,
         0, 0, 0, 1] #вращение вокруг OZ
    Y = [1, 0, 0, 0,
         0, cos(alpha), -sin(alpha), 0,
         0, sin(alpha), cos(alpha), 0,
         0, 0, 0, 1] #вращение вокруг OY
    glMultMatrixf(Z)
    glMultMatrixf(Y)
    cube(0.3) #основной куб

    glLoadIdentity()
    T2 = [1, 0, 0, 0,
          0, 1, 0, 0,
          0, 0, 1, 0,
          0.7, 0.8, 0, 1] #координаты маленького кубика
    glMultMatrixf(T2)
    cube(0.1) #маленький кубик
    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:
            beta += 0.1
        elif key == glfw.KEY_LEFT:
            beta -= 0.1
        elif key == glfw.KEY_UP:
            alpha += 0.1
        elif key == glfw.KEY_DOWN:
            alpha -= 0.1
        elif key == glfw.KEY_F:
            global fill
            if fill :
                glPolygonMode(GL_FRONT_AND_BACK, GL_FILL) #твердотельное
            else:
                glPolygonMode(GL_FRONT_AND_BACK, GL_LINE) #каркасное
            fill = not fill

def main():
    if not glfw.init():
        return
```

```
    window = glfw.create_window(640, 640, "laba2", None, None)
    if not window:
        glfw.terminate()
        return
    glfw.make_context_current(window)
    glfw.set_key_callback(window, key_callback)
    while not glfw.window_should_close(window):
        display(window)
    glfw.destroy_window(window)
    glfw.terminate()

main()
```

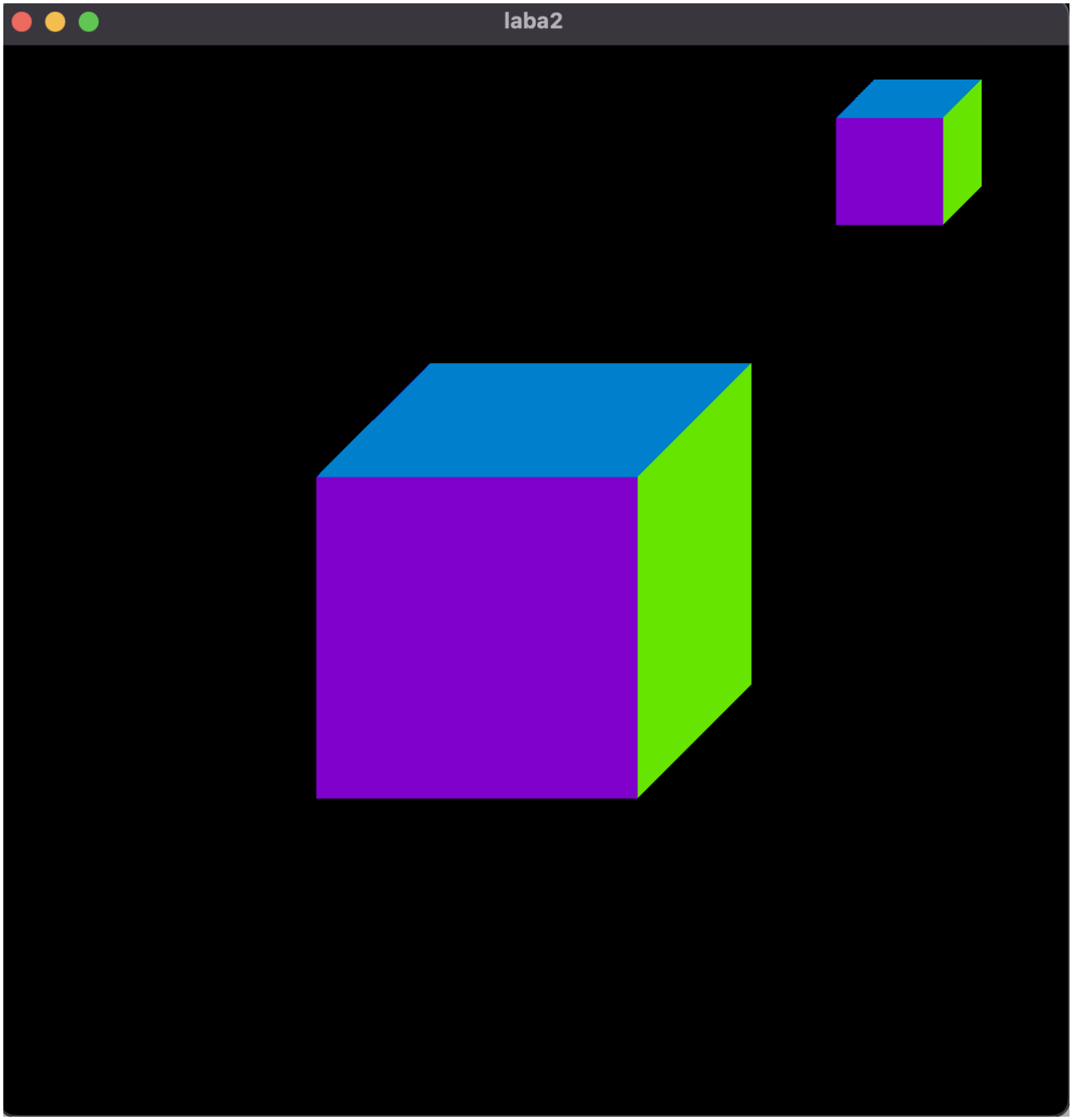

Рис. 1 — Начальное состояние

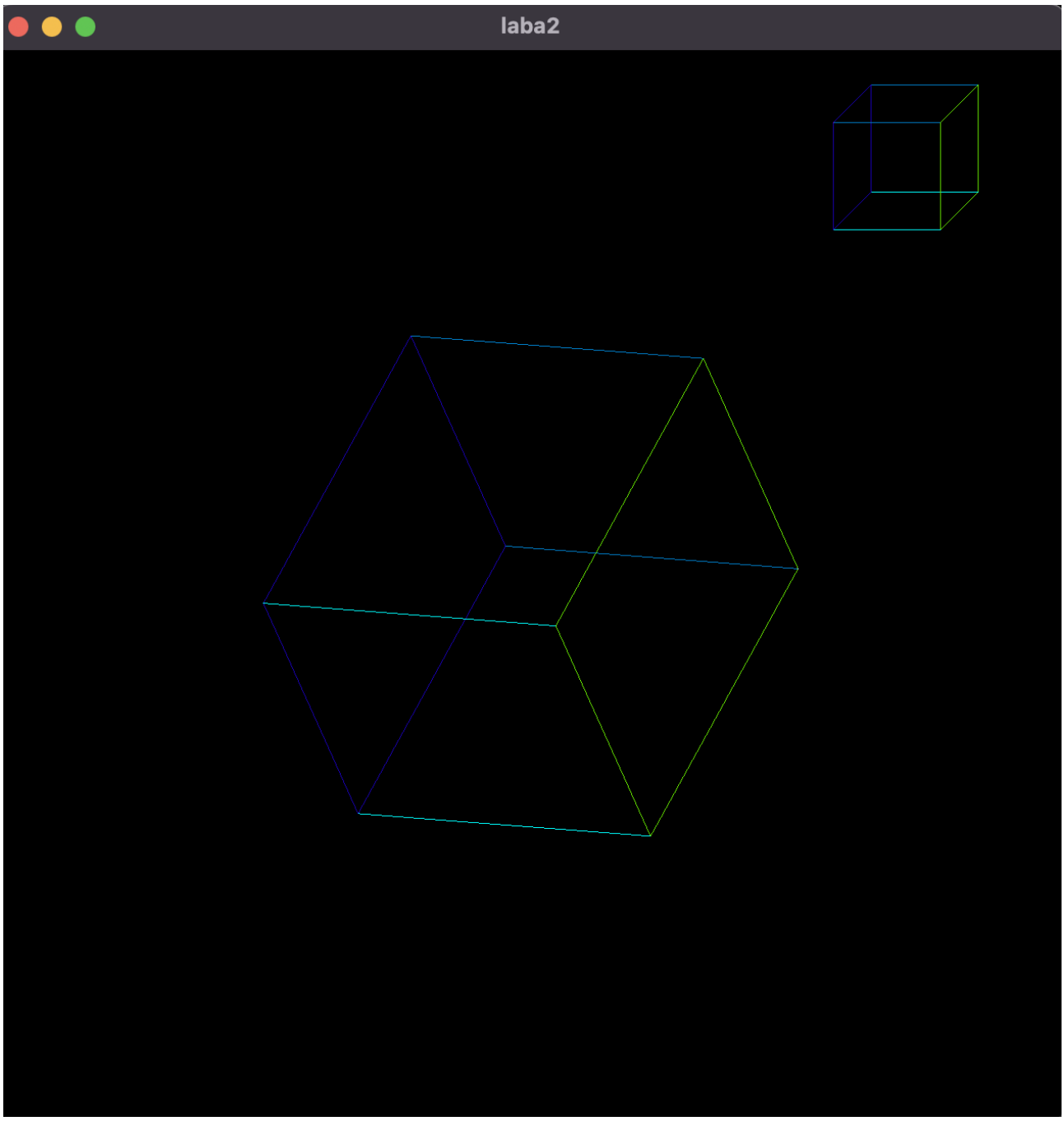

Рис. 2 — Каркасное отображение куба после поворотов

4 Заключение

Я узнала, что такое и из чего состоят модельно-видовые преобразования: преобразования координат моделей отдельных объектов, заданных в локальных системах координат, к мировой системе координат и учётом положения наблюдателя, состоят из композиции переносов, поворотов, масштабирований. Познакомилась с функциями операций над матрицами преобразований и видами матриц преобразований. Также я определила преобразования, позволяющие получить кабинетную проекцию (фронтальную диметрию) при помощи задания специальных матриц, реализовала изменение ориентации объекта с помощью модельно-видовых преобразований с возможностью управления при помощи клавиш и переключение между каркасным и твёрдотельным отображением модели.