ГУАП

КАФЕДРА № 42

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| канд. техн. наук, доцент |  |  |  | А. В. Аграновский |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 8 |
| ЗНАКОМСТВО С OPENGL |
| по курсу: |
| КОМПЬЮТЕРНАЯ ГРАФИКА |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ гр. № | 4326 |  |  |  | Г. С. Томчук |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

1. Цель работы

Цель работы: ознакомиться с основами использования библиотеки OpenGL для создания трёхмерных графических сцен, а также разработать динамическую 3D-сцену с использованием языка программирования высокого уровня, реализовав визуализацию, освещение и анимацию объектов.

1. Формулировка задания

Необходимо с использованием языка программирования высокого уровня (Python), поддерживающего библиотеку OpenGL, построить динамическую 3D сцену: добавить несколько трёхмерных объектов различных форм, анимации, освещение; обеспечить плавное изменение кадров и корректную обработку изменений размера окна.

1. Теоретические сведения

OpenGL (Open Graphics Library) — это открытый стандарт, предназначенный для разработки 2D и 3D компьютерной графики. Он обеспечивает набор функций, которые позволяют программистам взаимодействовать с графическим процессором (GPU) для создания графических объектов и сцен.

Основные возможности OpenGL:

* Построение геометрических фигур.
* Применение текстур и материалов.
* Работа с освещением и тенями.
* Анимация объектов.
* Обработка пользовательских событий.

OpenGL основан на концепции конвейера (pipeline), который состоит из нескольких этапов:

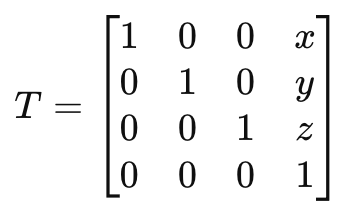
1. Построение геометрии сцены.
2. Применение трансформаций (перемещение, вращение, масштабирование).
3. Освещение и текстурирование.
4. Отображение готового изображения.

**glClear()** Очищает буфер экрана перед рендерингом нового кадра.

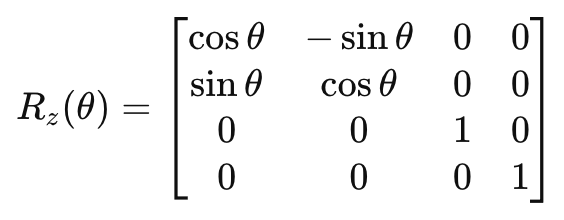
* GL\_COLOR\_BUFFER\_BIT — очищает буфер цвета.
* GL\_DEPTH\_BUFFER\_BIT — очищает буфер глубины (для корректного наложения объектов).

**glLoadIdentity()** Сбрасывает текущую матрицу на единичную. Используется перед применением трансформаций.

**glTranslatef(x, y, z)** Выполняет перенос объекта на заданное расстояние по координатам x, y, z. Матрица трансформации для переноса:



**glRotatef(angle, x, y, z)** Выполняет вращение объекта на заданный угол вокруг оси (x, y, z). Пример для вращения вокруг оси z:



**glColor3f(r, g, b)** Устанавливает цвет объекта, где r,g,br, g, br,g,b — значения в диапазоне [0, 1].

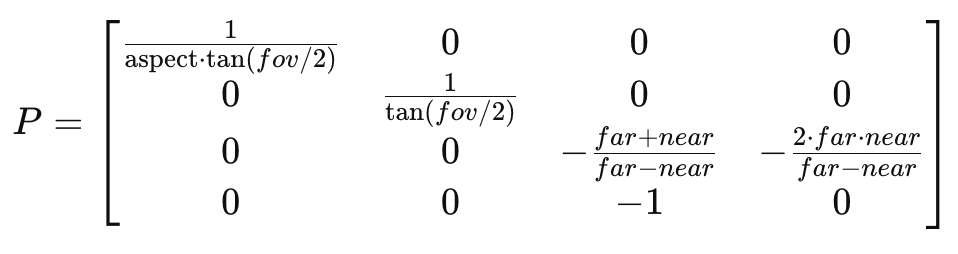
**glutSolidCube(size)**, **glutSolidSphere(radius, slices, stacks)** Рисуют стандартные трёхмерные объекты (куб, сфера) с заданными параметрами.

**glEnable()** и **glLightfv()** Настраивают освещение.

* GL\_LIGHTING включает освещение.
* GL\_LIGHT0 добавляет источник света.
* Функция glLightfv() задаёт параметры света (позицию, цвет и интенсивность).

**gluPerspective(fov, aspect, near, far)** Настраивает перспективную проекцию.

* fov — угол обзора.
* aspect — соотношение сторон окна.
* near, far — расстояния до ближней и дальней плоскостей отсечения.  
  Матрица перспективной проекции:



**glutTimerFunc()** Обновляет параметры анимации с заданным интервалом. В данной работе 16 мс — интервал между кадрами (примерно 60 FPS).

OpenGL использует модель освещения, основанную на линейной комбинации трёх составляющих:

1. **Фоновое освещение (Ambient):** равномерное освещение сцены.
2. **Диффузное освещение (Diffuse):** зависит от угла падения света на поверхность.
3. **Зеркальное освещение (Specular):** блеск на поверхности.
4. Листинг с кодом программы

from OpenGL.GL import \*

from OpenGL.GLU import \*

from OpenGL.GLUT import \*

*# Углы вращения для каждого объекта и группы*

angle\_group = 0.0

angle\_cube = 0.0

angle\_sphere = 0.0

angle\_pyramid = 0.0

*# Смещение куба вдоль Y*

y\_offset\_cube = -3.0

is\_y\_descend\_cube = False

*# Функция для настройки OpenGL*

def init():

glClearColor(0.1, 0.1, 0.1, 1.0) *# Темный фон*

glEnable(GL\_DEPTH\_TEST) *# Включить тест глубины*

glEnable(GL\_LIGHTING) *# Включить освещение*

glEnable(GL\_LIGHT0) *# Источник света 0*

glEnable(GL\_COLOR\_MATERIAL) *# Включить использование цвета материалов*

glColorMaterial(GL\_FRONT, GL\_AMBIENT\_AND\_DIFFUSE) *# Настройка материалов*

*# Настройка света*

light\_position = [1.0, 1.0, 1.0, 0.0]

light\_ambient = [0.2, 0.2, 0.2, 1.0]

light\_diffuse = [0.8, 0.8, 0.8, 1.0]

light\_specular = [1.0, 1.0, 1.0, 1.0]

glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position)

glLightfv(GL\_LIGHT0, GL\_AMBIENT, light\_ambient)

glLightfv(GL\_LIGHT0, GL\_DIFFUSE, light\_diffuse)

glLightfv(GL\_LIGHT0, GL\_SPECULAR, light\_specular)

*# Функция для отображения сцены*

def display():

global angle\_group

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT) *# Очистка экрана и буфера глубины*

glLoadIdentity() *# Сброс матрицы*

*# Перемещаем камеру*

glTranslatef(0.0, 0.0, -15.0)

*# Групповое вращение всех объектов*

glRotatef(angle\_group, 0.0, 1.0, 0.0)

*# Рисуем объекты*

draw\_cube(angle\_cube, y\_offset\_cube)

draw\_sphere(angle\_sphere)

draw\_pyramid(angle\_pyramid)

glutSwapBuffers() *# Меняем буферы для двойной буферизации*

*# Функция для рисования куба*

def draw\_cube(angle, y\_offset):

glPushMatrix()

glTranslatef(-3.0, y\_offset, 0.0) *# Смещение куба влево*

glRotatef(angle, 1.0, 1.0, 0.0) *# Вращение куба вокруг собственной оси*

glColor3f(0.0, 1.0, 0.0) *# Зеленый цвет*

glutSolidCube(2.0) *# Рисуем куб размером 2*

glPopMatrix()

*# Функция для рисования сферы*

def draw\_sphere(angle):

glPushMatrix()

glTranslatef(3.0, 0.0, 0.0) *# Смещение сферы вправо*

glRotatef(angle, 0.0, 1.0, 0.0) *# Вращение сферы вокруг своей оси*

glColor3f(1.0, 0.0, 0.0) *# Красный цвет*

glutSolidSphere(1.5, 50, 50) *# Рисуем сферу радиусом 1.5*

glPopMatrix()

*# Функция для рисования пирамиды*

def draw\_pyramid(angle):

glPushMatrix()

glTranslatef(0.0, -3.0, 0.0) *# Смещение пирамиды вниз*

glRotatef(angle, 0.0, 1.0, 0.0) *# Вращение пирамиды вокруг своей оси*

glColor3f(0.0, 0.0, 1.0) *# Синий цвет*

glBegin(GL\_TRIANGLES)

*# Передняя грань*

glVertex3f(0.0, 2.0, 0.0)

glVertex3f(-1.0, 0.0, 1.0)

glVertex3f(1.0, 0.0, 1.0)

*# Правая грань*

glVertex3f(0.0, 2.0, 0.0)

glVertex3f(1.0, 0.0, 1.0)

glVertex3f(1.0, 0.0, -1.0)

*# Задняя грань*

glVertex3f(0.0, 2.0, 0.0)

glVertex3f(1.0, 0.0, -1.0)

glVertex3f(-1.0, 0.0, -1.0)

*# Левая грань*

glVertex3f(0.0, 2.0, 0.0)

glVertex3f(-1.0, 0.0, -1.0)

glVertex3f(-1.0, 0.0, 1.0)

glEnd()

glPopMatrix()

*# Функция для обработки изменения размера окна*

def reshape(width, height):

if height == 0:

height = 1

aspect = width / height

glViewport(0, 0, width, height)

glMatrixMode(GL\_PROJECTION)

glLoadIdentity()

gluPerspective(45.0, aspect, 0.1, 50.0)

glMatrixMode(GL\_MODELVIEW)

glLoadIdentity()

*# Функция для обновления углов вращения*

def update(value):

global angle\_group, angle\_cube, angle\_sphere, angle\_pyramid, y\_offset\_cube, is\_y\_descend\_cube

*# Обновляем углы вращения*

angle\_group += 1.0

angle\_cube += 2.0

angle\_sphere += 1.5

angle\_pyramid += 1.0

if angle\_group > 360:

angle\_group -= 360

if angle\_cube > 360:

angle\_cube -= 360

if angle\_sphere > 360:

angle\_sphere -= 360

if angle\_pyramid > 360:

angle\_pyramid -= 360

*# Обновляем смещение куба по Y*

if y\_offset\_cube >= 3.0:

is\_y\_descend\_cube = True

if y\_offset\_cube <= -3.0:

is\_y\_descend\_cube = False

if is\_y\_descend\_cube:

y\_offset\_cube -= 0.05

else:

y\_offset\_cube += 0.05

glutPostRedisplay()

glutTimerFunc(16, update, 0)

*# Главная функция*

def main():

glutInit()

glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH)

glutInitWindowSize(800, 600)

glutCreateWindow("3D Сцена")

glutDisplayFunc(display)

glutReshapeFunc(reshape)

glutTimerFunc(0, update, 0)

init()

glutMainLoop()

if \_\_name\_\_ == "\_\_main\_\_":

main()

1. Экранные формы с результатами работы программы

На рис. 1, 2, 3 представлены скриншоты окна программы в произвольные моменты времени. Фигуры вращаются отдельно вдоль собственных осей, вращаются как отдельная группа объектов, а также куб смещается вверх-вниз по оси Y.

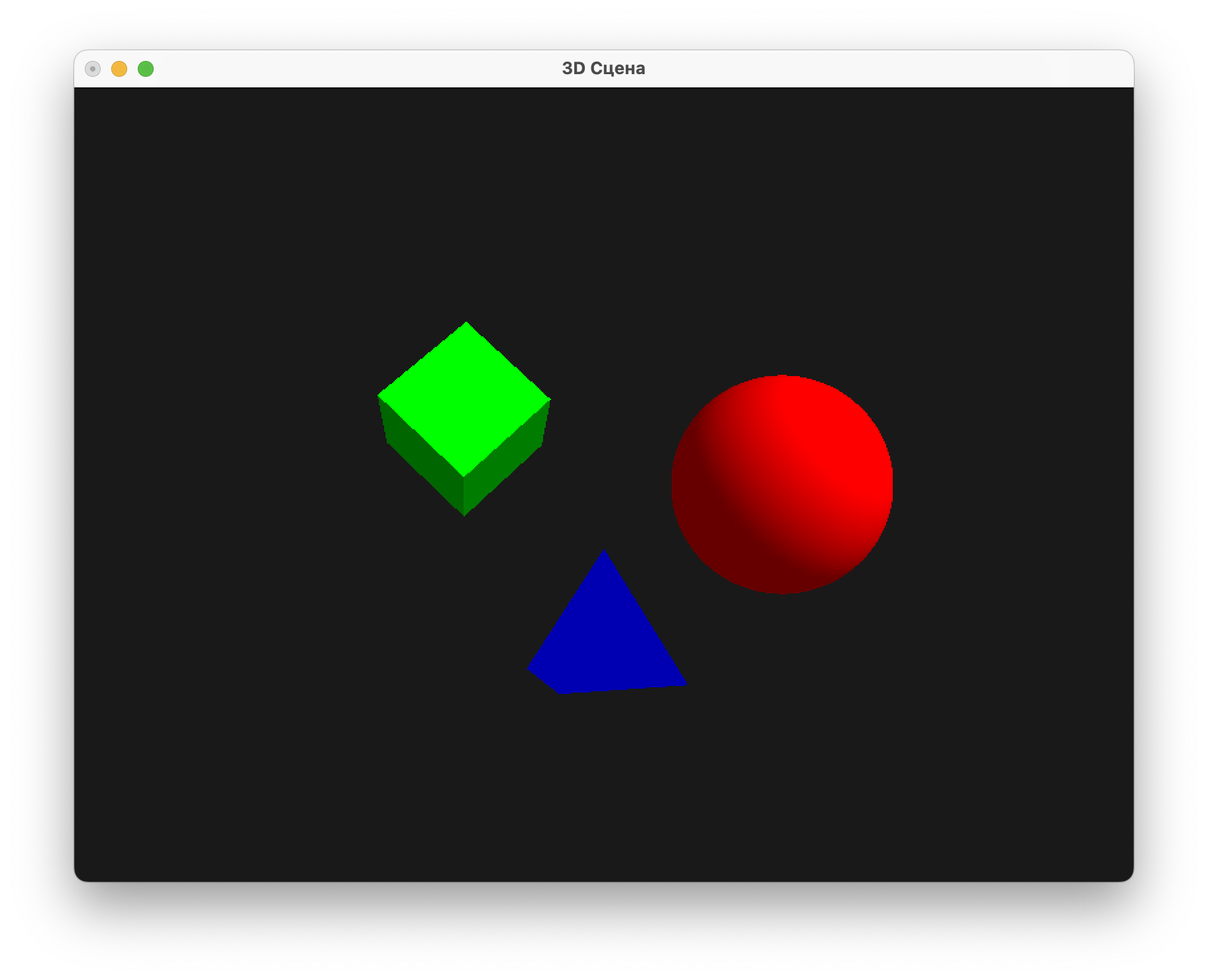


Рисунок 1

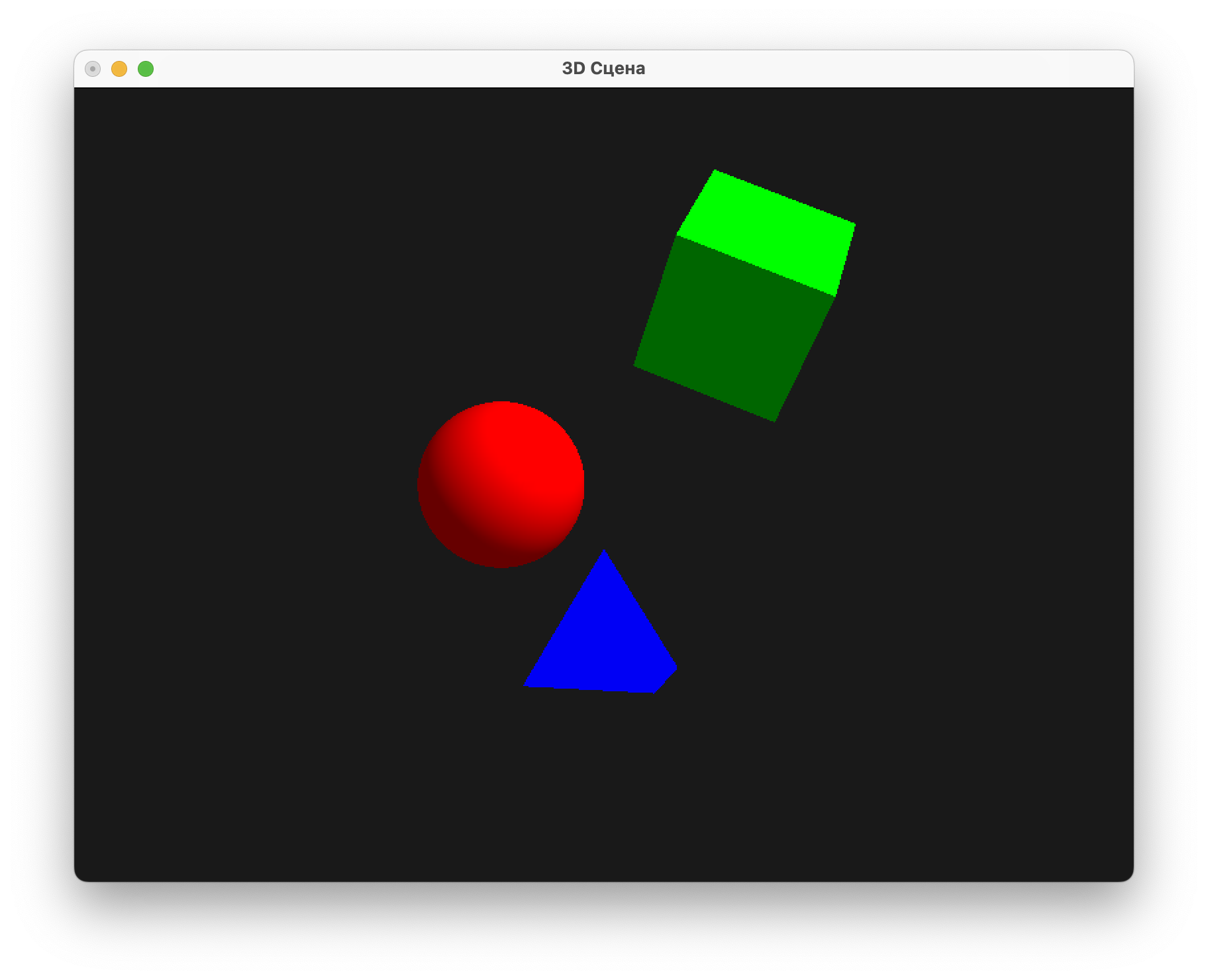


Рисунок 2

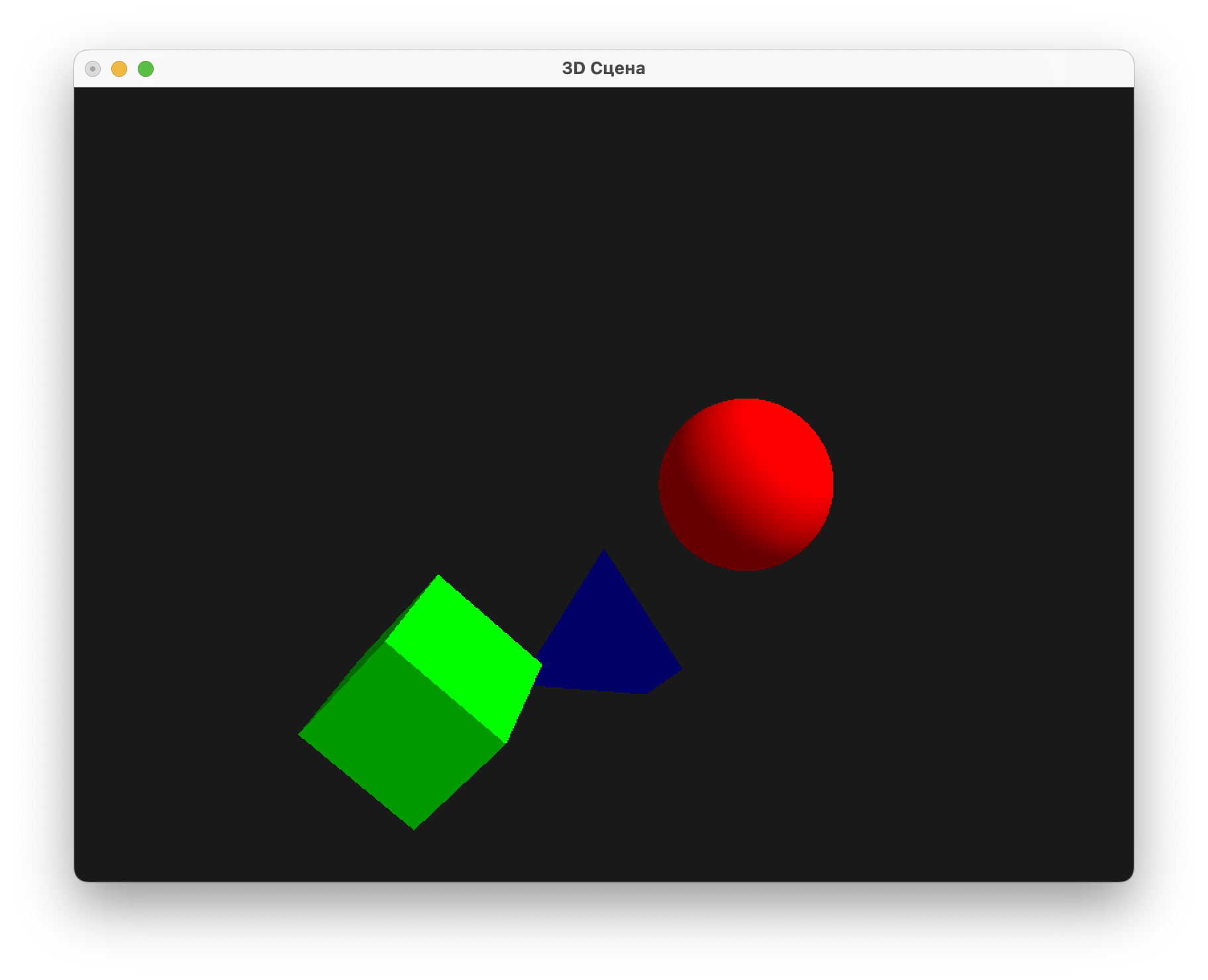


Рисунок 3

1. Выводы

В ходе выполнения лабораторной работы были изучены основы использования библиотеки OpenGL для создания трёхмерных графических сцен. В результате:

* + - 1. Реализована динамическая 3D-сцена с несколькими геометрическими объектами, которые анимированы как по отдельности (вращение вокруг своей оси), так и группой (вращение вокруг общей оси).
      2. Добавлено освещение, что улучшило визуальное восприятие сцены, подчеркнув объёмность объектов и их взаимодействие со светом.
      3. Закреплены навыки работы с основными функциями OpenGL, такими как настройка проекции, управление трансформациями (перемещение, вращение), а также настройка освещения.
      4. Использованы базовые математические операции над матрицами, которые лежат в основе всех трансформаций в OpenGL.

Данная работа позволила лучше понять принципы работы графического конвейера, основы работы с трехмерной графикой и научиться создавать простые анимации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Петров, А. Н., Смирнов, И. В. Основы программирования на Python: Учебное пособие. – Санкт-Петербург: Издательство СПб ГУАП, 2022. – 320 с.
2. Бобылев, С. И. Основы компьютерной графики: учебное пособие / С. И. Бобылев. — М.: Издательство МГТУ им. Н. Э. Баумана, 2010. — 224 с.
3. Райт, Р.С.-мл., Липчак Б. OpenGL. Суперкнига, 3-е издание. — М.: Издательский дом «Вильямс», 2006. - 1040 с.