Коба Алексей Юрьевич  
гр. 5130904/20101  
Вариант № 10

**Практическая работа №1**

**Текст задания:**

1. Изобразить каркасный икосаэдр и каркасный конус. Размеры примитивов задать самостоятельно.
2. Выполнить поворот конуса на α = -60 градусов вокруг оси X, сдвиг икосаэдра по оси Z на произвольное расстояние.
3. Изобразить чайник и тор произвольных размеров.
4. Промасштабировать тор с коэффициентом 0.5.

**Описание хода работы**

**1. Инициализация OpenGL**

Для выполнения работы использовалась библиотека PyOpenGL, предоставляющая доступ к стандартным функциям OpenGL. Была выполнена инициализация графического окна размером 1200×800 пикселей с двойной буферизацией (GLUT\_DOUBLE) и тестом глубины (GL\_DEPTH\_TEST) для корректного отображения 3D объектов. Настроена перспективная проекция с углом обзора 45° с помощью gluPerspective().

**2. Реализация задания 1**

С использованием стандартных функций GLUT были созданы каркасные примитивы на одной сцене:

* glutWireIcosahedron() – для отрисовки икосаэдра (многогранник с 20 гранями);
* glutWireCone(1.2, 2.5, 20, 20) – для отрисовки конуса с радиусом основания 1.2 и высотой 2.5.

Икосаэдр увеличен в 1.5 раза с помощью glScalef(1.5, 1.5, 1.5) и смещён влево (glTranslatef(-2.5, 0.0, 0.0)). Конус смещён вправо (glTranslatef(2.5, 0.0, 0.0)). Для визуализации системы координат отрисованы оси: X (красная), Y (зелёная), Z (синяя).

**3. Реализация задания 2**

Применены трансформации к объектам из задания 1:

* Конус повёрнут на -60° вокруг оси X с использованием glRotatef(-60.0, 1.0, 0.0, 0.0);
* Икосаэдр смещён по оси Z на 3.0 единицы с помощью glTranslatef(-2.5, 0.0, 3.0).

Оба объекта отображены на одной сцене для демонстрации эффекта трансформаций.

**4. Реализация задания 3**

Созданы каркасные примитивы на одной сцене:

* glutWireTeapot(1.2) – чайник с размером 1.2, смещённый влево и вниз (glTranslatef(-2.5, -0.5, 0.0));
* glutWireTorus(0.5, 1.5, 20, 30) – тор с внутренним радиусом 0.5 и внешним радиусом 1.5, смещённый вправо (glTranslatef(2.5, 0.0, 0.0)).

**5. Реализация задания 4**

Выполнено масштабирование тора с коэффициентом 0.5 с использованием glScalef(0.5, 0.5, 0.5). На одной сцене отображены:

* Чайник (слева) – без изменений для сравнения;
* Масштабированный тор (справа) – уменьшен в 2 раза по всем осям.

Это позволяет наглядно продемонстрировать эффект масштабирования.

**6. Интерактивное управление**

Реализована орбитальная камера с управлением через мышь и клавиатуру:

* Вращение сцены – удержание левой кнопки мыши (ЛКМ) и перемещение курсора. Вращение реализовано в сферических координатах с помощью gluLookAt();
* Зум (приближение/отдаление) – прокрутка колесика мыши (изменение параметра zoom);
* Переключение между заданиями – клавиши 1, 2, 3, 4;
* Сброс камеры – клавиша R возвращает камеру к начальным углам (rotation\_x=30°, rotation\_y=45°);
* Сохранение скриншота – клавиша S;
* Выход – клавиша ESC.

**7. Автоматическое сохранение изображений**

При переключении между заданиями программа автоматически сохраняет скриншоты в формате PNG для отражения в отчете (см. ниже). Использована библиотека Pillow для захвата содержимого буфера кадра OpenGL.

**Результаты выполнения программы**

**Задание 1. Каркасный икосаэдр и каркасный конус**

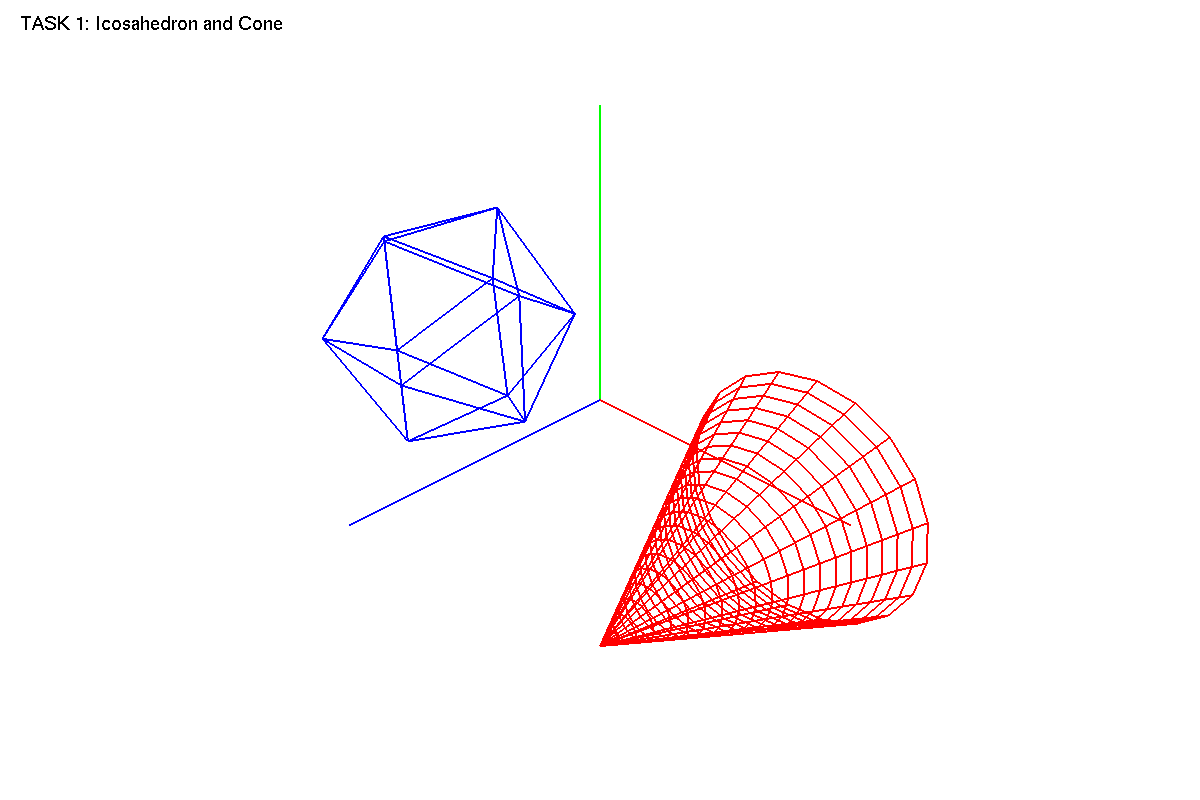
**

Рисунок 1. Каркасный икосаэдр (слева) и каркасный конус (справа). Отображены оси координат: X (красная), Y (зеленая), Z (синяя).

**Задание 2. Трансформации объектов**

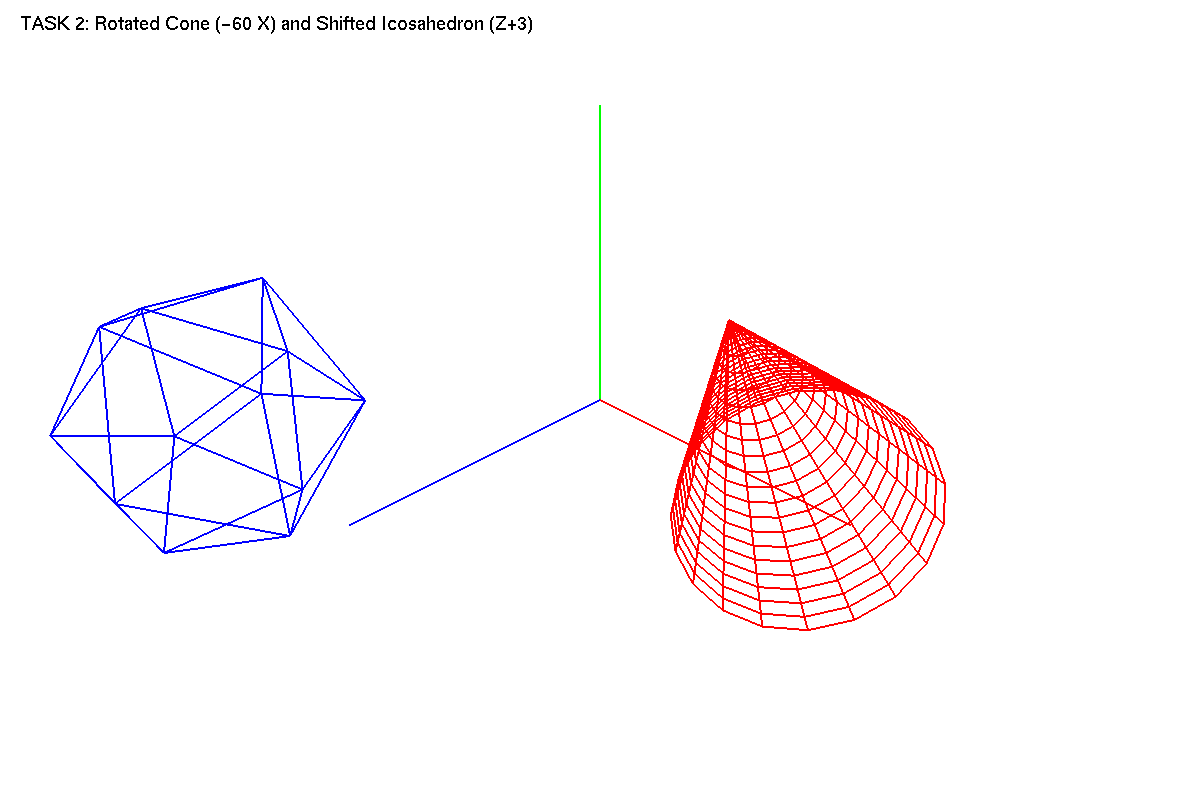
**

Рисунок 2. Икосаэдр со сдвигом по оси Z на 3.0 единицы (слева) и конус с поворотом на -60° вокруг оси X (справа).

**Задание 3. Каркасный чайник и каркасный тор**

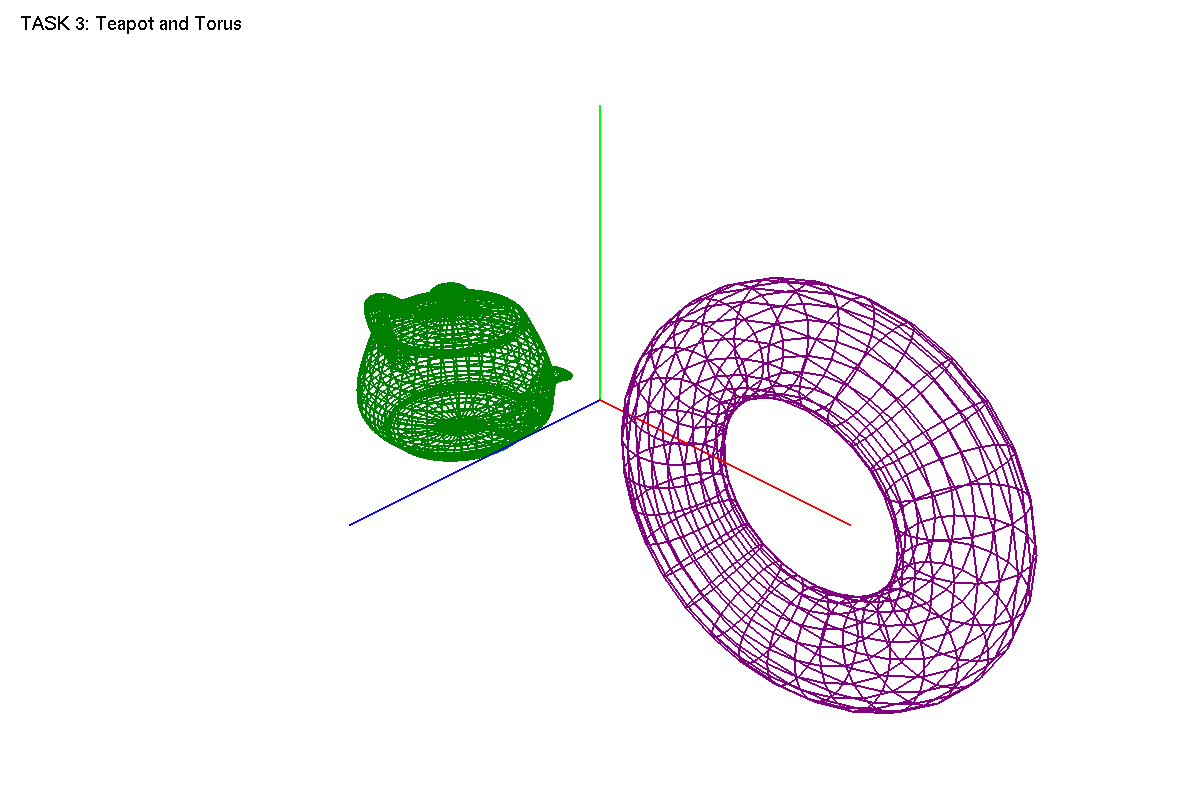
**

Рисунок 3. Каркасный чайник Юты (слева) и каркасный тор (справа).

**Задание 4. Масштабирование тора**

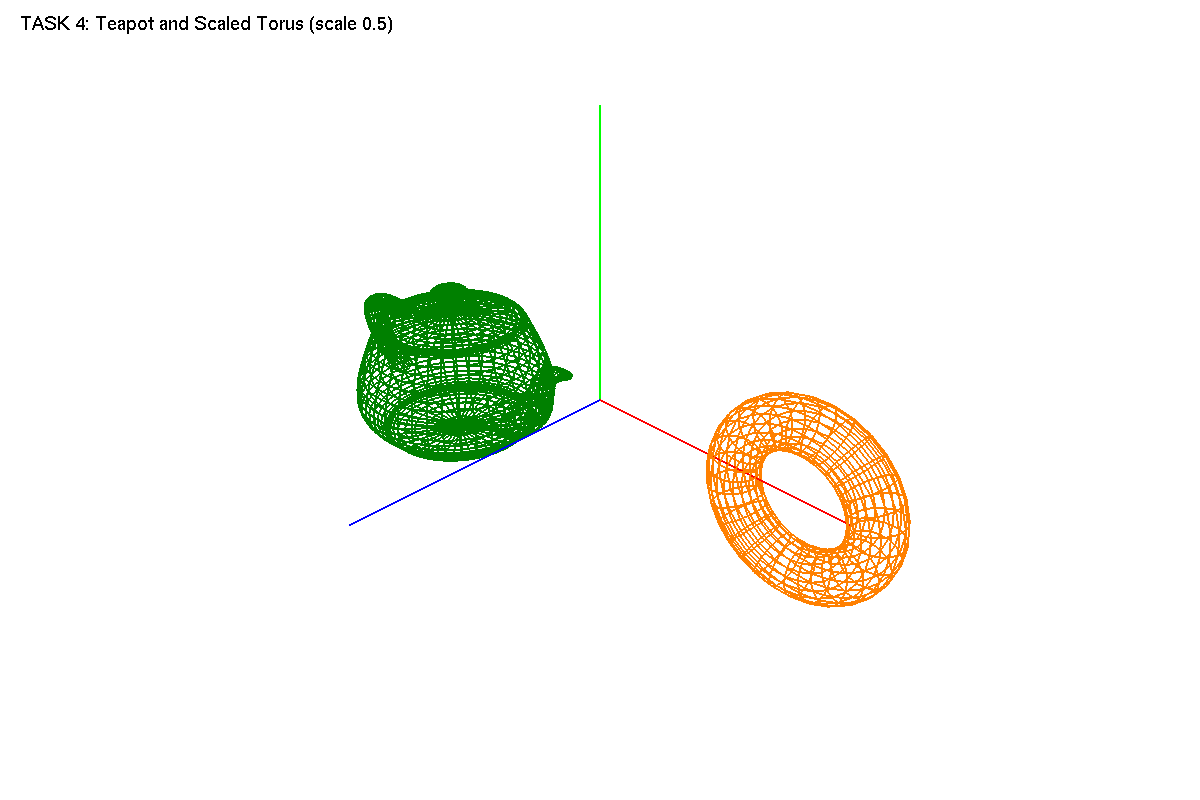
**

Рисунок 4. Исходный тор (слева) и тор с масштабом 0.5 (справа).

**ПРИЛОЖЕНИЕ**

**Листинг программы**

| from OpenGL.GL import \* from OpenGL.GLU import \* from OpenGL.GLUT import \* import sys import numpy as np from PIL import Image  window\_width = 1200 window\_height = 800 current\_task = 1 screenshot\_taken = [False, False, False, False]  mouse\_down = False mouse\_x = 0 mouse\_y = 0 rotation\_x = 30.0 rotation\_y = 45.0 zoom = 1.0  def init\_opengl():  """Инициализация параметров OpenGL"""  glClearColor(1.0, 1.0, 1.0, 1.0)  glEnable(GL\_DEPTH\_TEST)  glMatrixMode(GL\_PROJECTION)  glLoadIdentity()  gluPerspective(45, window\_width / window\_height, 0.1, 50.0)  glMatrixMode(GL\_MODELVIEW)  def draw\_axes():  """Отрисовка осей координат X, Y, Z"""  glLineWidth(2.0)   # Ось X (красная)  glColor3f(1.0, 0.0, 0.0)  glBegin(GL\_LINES)  glVertex3f(0.0, 0.0, 0.0)  glVertex3f(3.0, 0.0, 0.0)  glEnd()   # Ось Y (зеленая)  glColor3f(0.0, 1.0, 0.0)  glBegin(GL\_LINES)  glVertex3f(0.0, 0.0, 0.0)  glVertex3f(0.0, 3.0, 0.0)  glEnd()   # Ось Z (синяя)  glColor3f(0.0, 0.0, 1.0)  glBegin(GL\_LINES)  glVertex3f(0.0, 0.0, 0.0)  glVertex3f(0.0, 0.0, 3.0)  glEnd()  def draw\_text(x, y, text):  glDisable(GL\_DEPTH\_TEST)   glMatrixMode(GL\_PROJECTION)  glPushMatrix()  glLoadIdentity()  gluOrtho2D(0, window\_width, 0, window\_height)   glMatrixMode(GL\_MODELVIEW)  glPushMatrix()  glLoadIdentity()   glColor3f(0.0, 0.0, 0.0)  glRasterPos2f(x, y)  for char in text:  glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_18, ord(char))   glPopMatrix()  glMatrixMode(GL\_PROJECTION)  glPopMatrix()   glMatrixMode(GL\_MODELVIEW)  glEnable(GL\_DEPTH\_TEST)  def save\_screenshot(filename):  glReadBuffer(GL\_FRONT)  pixels = glReadPixels(0, 0, window\_width, window\_height, GL\_RGB, GL\_UNSIGNED\_BYTE)  image = Image.frombytes("RGB", (window\_width, window\_height), pixels)  image = image.transpose(Image.FLIP\_TOP\_BOTTOM)  image.save(filename)  print(f"✓ Сохранено изображение: {filename}")  def apply\_camera\_rotation():  """Применение вращения камеры на основе позиции мыши"""  distance = 10.0 \* zoom   rot\_x\_rad = np.radians(rotation\_x)  rot\_y\_rad = np.radians(rotation\_y)   cam\_x = distance \* np.sin(rot\_y\_rad) \* np.cos(rot\_x\_rad)  cam\_y = distance \* np.sin(rot\_x\_rad)  cam\_z = distance \* np.cos(rot\_y\_rad) \* np.cos(rot\_x\_rad)   return cam\_x, cam\_y, cam\_z  def display\_task\_1():  """ЗАДАНИЕ 1: Каркасный икосаэдр и каркасный конус на одной плоскости"""  glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)  glLoadIdentity()   cam\_x, cam\_y, cam\_z = apply\_camera\_rotation()  gluLookAt(cam\_x, cam\_y, cam\_z, 0, 0, 0, 0, 1, 0)   draw\_axes()   # Икосаэдр слева  glPushMatrix()  glTranslatef(-2.5, 0.0, 0.0)  glColor3f(0.0, 0.0, 1.0)  glScalef(1.5, 1.5, 1.5)  glutWireIcosahedron()  glPopMatrix()   # Конус справа  glPushMatrix()  glTranslatef(2.5, 0.0, 0.0)  glColor3f(1.0, 0.0, 0.0)  glutWireCone(1.2, 2.5, 20, 20)  glPopMatrix()   draw\_text(20, window\_height - 30, "TASK 1: Icosahedron and Cone")   glutSwapBuffers()   global screenshot\_taken  if not screenshot\_taken[0]:  save\_screenshot("zadanie\_1\_opengl.png")  screenshot\_taken[0] = True  def display\_task\_2():  """ЗАДАНИЕ 2: Поворот конуса на -60° вокруг оси X, сдвиг икосаэдра по Z"""  glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)  glLoadIdentity()   cam\_x, cam\_y, cam\_z = apply\_camera\_rotation()  gluLookAt(cam\_x, cam\_y, cam\_z, 0, 0, 0, 0, 1, 0)   draw\_axes()   # Икосаэдр со сдвигом по Z  glPushMatrix()  glTranslatef(-2.5, 0.0, 3.0) # Сдвиг по Z на 3.0  glColor3f(0.0, 0.0, 1.0)  glScalef(1.5, 1.5, 1.5)  glutWireIcosahedron()  glPopMatrix()   # Конус с поворотом на -60° вокруг оси X  glPushMatrix()  glTranslatef(2.5, 0.0, 0.0)  glRotatef(-60.0, 1.0, 0.0, 0.0) # Поворот на -60° вокруг оси X  glColor3f(1.0, 0.0, 0.0)  glutWireCone(1.2, 2.5, 20, 20)  glPopMatrix()   draw\_text(20, window\_height - 30, "TASK 2: Rotated Cone (-60 X) and Shifted Icosahedron (Z+3)")   glutSwapBuffers()   global screenshot\_taken  if not screenshot\_taken[1]:  save\_screenshot("zadanie\_2\_opengl.png")  screenshot\_taken[1] = True  def display\_task\_3():  """ЗАДАНИЕ 3: Каркасный чайник и каркасный тор на одной плоскости"""  glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)  glLoadIdentity()   cam\_x, cam\_y, cam\_z = apply\_camera\_rotation()  gluLookAt(cam\_x, cam\_y, cam\_z, 0, 0, 0, 0, 1, 0)   draw\_axes()   # Чайник слева  glPushMatrix()  glTranslatef(-2.5, -0.5, 0.0)  glColor3f(0.0, 0.5, 0.0)  glutWireTeapot(1.2)  glPopMatrix()   # Тор справа  glPushMatrix()  glTranslatef(2.5, 0.0, 0.0)  glColor3f(0.5, 0.0, 0.5)  glutWireTorus(0.5, 1.5, 20, 30)  glPopMatrix()   draw\_text(20, window\_height - 30, "TASK 3: Teapot and Torus")   glutSwapBuffers()   global screenshot\_taken  if not screenshot\_taken[2]:  save\_screenshot("zadanie\_3\_opengl.png")  screenshot\_taken[2] = True  def display\_task\_4():  """ЗАДАНИЕ 4: Чайник и тор с коэффициентом масштабирования 0.5"""  glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)  glLoadIdentity()   cam\_x, cam\_y, cam\_z = apply\_camera\_rotation()  gluLookAt(cam\_x, cam\_y, cam\_z, 0, 0, 0, 0, 1, 0)   draw\_axes()   # Чайник слева (без изменений)  glPushMatrix()  glTranslatef(-2.5, -0.5, 0.0)  glColor3f(0.0, 0.5, 0.0)  glutWireTeapot(1.2)  glPopMatrix()   # Тор справа с масштабированием 0.5  glPushMatrix()  glTranslatef(2.5, 0.0, 0.0)  glScalef(0.5, 0.5, 0.5) # Масштабирование с коэффициентом 0.5  glColor3f(1.0, 0.5, 0.0)  glutWireTorus(0.5, 1.5, 20, 30)  glPopMatrix()   draw\_text(20, window\_height - 30, "TASK 4: Teapot and Scaled Torus (scale 0.5)")   glutSwapBuffers()   global screenshot\_taken  if not screenshot\_taken[3]:  save\_screenshot("zadanie\_4\_opengl.png")  screenshot\_taken[3] = True  print("\n" + "="\*70)  print("✓ ВСЕ ИЗОБРАЖЕНИЯ СОХРАНЕНЫ!")  print("="\*70)  def display():  """Основная функция отображения"""  if current\_task == 1:  display\_task\_1()  elif current\_task == 2:  display\_task\_2()  elif current\_task == 3:  display\_task\_3()  elif current\_task == 4:  display\_task\_4()  def keyboard(key, x, y):  """Обработка нажатий клавиш"""  global current\_task, rotation\_x, rotation\_y, zoom   if key == b'1':  current\_task = 1  glutPostRedisplay()  elif key == b'2':  current\_task = 2  glutPostRedisplay()  elif key == b'3':  current\_task = 3  glutPostRedisplay()  elif key == b'4':  current\_task = 4  glutPostRedisplay()  elif key == b's':  save\_screenshot(f"screenshot\_task\_{current\_task}.png")  elif key == b'r':  rotation\_x = 30.0  rotation\_y = 45.0  zoom = 1.0  glutPostRedisplay()  print("Вращение сброшено к начальным значениям")  elif key == b'\x1b': # ESC  sys.exit(0)  def mouse(button, state, x, y):  """Обработка нажатий кнопок мыши"""  global mouse\_down, mouse\_x, mouse\_y, zoom   if button == GLUT\_LEFT\_BUTTON:  if state == GLUT\_DOWN:  mouse\_down = True  mouse\_x = x  mouse\_y = y  else:  mouse\_down = False   elif button == 3:  zoom \*= 0.9  glutPostRedisplay()  elif button == 4:  zoom \*= 1.1  glutPostRedisplay()  def motion(x, y):  """Обработка движения мыши с нажатой кнопкой"""  global mouse\_x, mouse\_y, rotation\_x, rotation\_y   if mouse\_down:  dx = x - mouse\_x  dy = y - mouse\_y   rotation\_y += dx \* 0.5  rotation\_x += dy \* 0.5   if rotation\_x > 89.0:  rotation\_x = 89.0  if rotation\_x < -89.0:  rotation\_x = -89.0   mouse\_x = x  mouse\_y = y   glutPostRedisplay()  def main():  glutInit(sys.argv)  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH)  glutInitWindowSize(window\_width, window\_height)  glutInitWindowPosition(100, 100)  glutCreateWindow(b"Lab1: OpenGL - Interactive")   init\_opengl()   glutDisplayFunc(display)  glutKeyboardFunc(keyboard)  glutMouseFunc(mouse)  glutMotionFunc(motion)   print("="\*70)  print("Лабораторная работа 1 - Исправленная версия")  print("="\*70)  print("\nУправление клавиатурой:")  print(" '1' - Задание 1: Икосаэдр и конус")  print(" '2' - Задание 2: Трансформации")  print(" '3' - Задание 3: Чайник и тор")  print(" '4' - Задание 4: Масштабирование тора")  print(" 'S' - Сохранить текущий кадр")  print(" 'R' - Сбросить вращение к начальным значениям")  print(" 'ESC' - Выход")  print("\nУправление мышью:")  print(" ЛКМ + движение мыши - Вращение сцены")  print(" Колесико мыши - Приближение/отдаление (зум)")  print("="\*70)  print("\nИзображения будут автоматически сохранены при переключении заданий.")  print("="\*70)   glutMainLoop()  if \_\_name\_\_ == "\_\_main\_\_":  main() |
| --- |