Коба Алексей Юрьевич  
гр. 5130904/20101  
Вариант № 10

**Практическая работа №2**

**Текст задания:**

Целью работы является ознакомление с основными функциями API, описывающими свойства материалов объектов и позволяющими задавать параметры источника освещения.

Требуется разработать программу, изображающую заданный набор из трех предметов с указанными свойствами материалов и параметры источника освещения. При этом в качестве базового набора объектов выступают 3D примитивы, указанные в вашем варианте задания №1. Следует наделить один из объектов свойствами прозрачности (значение параметра должно быть выше 0,5). Другой выбранный объект должен имитировать отполированную поверхность (shininess, значение указывается максимальным). В качестве такого объекта следует выбирать примитивы с выпуклыми поверхностями, например - цилиндр, тор, конус, сферу, чайник. Третий объект должен быть диффузно-рассеивающим, матовым.

В сцене обязательно должен быть как минимум один источник освещения, с возможностью менять его параметры: местоположение, интенсивность, цвет освещения.

Окончательный этап – текстурирование одного из объектов (матового). Возможно при этом также использовать микроискажение нормалей при помощи bump-mapping

**Описание хода работы**

**1. Инициализация OpenGL и настройка освещения**

Для выполнения работы использовалась библиотека PyOpenGL. Было инициализировано графическое окно размером 1400×900 пикселей с поддержкой освещения, прозрачности и текстурирования. Включены:

* Освещение (GL\_LIGHTING, GL\_LIGHT0) – для реалистичного отображения материалов;
* Буфер глубины (GL\_DEPTH\_TEST) – для корректной отрисовки перекрывающихся объектов;
* Alpha-blending (GL\_BLEND, GL\_SRC\_ALPHA) – для поддержки прозрачности.

**2. Реализация материалов объектов**

В соответствии с заданием созданы три объекта из варианта №10 (икосаэдр, чайник, тор) с различными свойствами материалов с использованием функции glMaterialfv():

**Объект 1: Прозрачный икосаэдр**

* Alpha-канал: 0.3 (70% прозрачности)
* Материал: ambient/diffuse/specular с alpha=0.3, shininess=30
* Для корректной прозрачности отключена запись в буфер глубины (glDepthMask(GL\_FALSE)) во время отрисовки
* Расположен слева, за ним виден чайник

**Объект 2: Отполированный чайник**

* Shininess: 128 (максимальное значение в OpenGL)
* Материал: ярко-фиолетовый diffuse (1.0, 0.0, 1.0), высокий specular (1.0, 1.0, 1.0)
* Имитирует зеркальную отполированную поверхность с яркими бликами
* Расположен за прозрачным икосаэдром для демонстрации эффекта прозрачности

**Объект 3: Матовый тор с текстурой**

* Shininess: 5 (низкий блеск)
* Материал: серый diffuse (0.8, 0.8, 0.8), минимальный specular (0.1, 0.1, 0.1)
* Имитирует матовую диффузно-рассеивающую поверхность без бликов

**3. Настройка источника освещения**

Реализован точечный источник света (GL\_LIGHT0) с интерактивным управлением параметрами:

* Позиция: glLightfv(GL\_LIGHT0, GL\_POSITION, [x, y, z, 1.0]) – четвёртый параметр 1.0 задаёт точечный источник
* Компоненты освещения:
  + GL\_AMBIENT – фоновое освещение (0.2 × интенсивность)
  + GL\_DIFFUSE – направленный свет с учётом цвета и интенсивности
  + GL\_SPECULAR – белый свет для бликов (1.0, 1.0, 1.0)
* Визуальный маркер источника света (жёлтая каркасная сфера) отображается в сцене

**4. Текстурирование матового объекта**

На матовый тор применена процедурная текстура (шахматная доска 256×256 пикселей):

* Текстура генерируется программно с помощью NumPy (чередование оранжевых и синих квадратов)
* Загружена в OpenGL с помощью glTexImage2D() и параметров GL\_REPEAT, GL\_LINEAR
* Для GLUT-примитива glutSolidTorus используется автоматическая генерация текстурных координат (GL\_TEXTURE\_GEN\_S/T с режимом GL\_OBJECT\_LINEAR), так как GLUT не предоставляет встроенные UV-координаты

**5. Корректная отрисовка прозрачности**

Для правильного отображения прозрачного икосаэдра соблюдён порядок отрисовки:

1. Непрозрачные объекты (тор, чайник)
2. Прозрачные объекты (икосаэдр) – рисуются последними с отключённой записью в Z-buffer

Это гарантирует, что объекты позади прозрачного икосаэдра (чайник) будут видны сквозь него.

**6. Интерактивное управление**

Реализовано управление с помощью клавиатуры и мыши:

Управление освещением:

* W/A/S/D/Q/E – перемещение источника света по осям X, Y, Z
* +/- – изменение интенсивности освещения (0.0 – 2.0)
* 1-5 – изменение цвета света (красный, зелёный, синий, жёлтый, белый)

Управление камерой:

* ЛКМ + движение мыши – вращение орбитальной камеры вокруг сцены
* Колесико мыши – зум (приближение/отдаление)
* R – сброс камеры к начальным значениям

Прочее:

* T – переключение текстуры тора (вкл/выкл)
* H – вывод справки по управлению
* ESC – выход из программы

**Результаты выполнения программы**

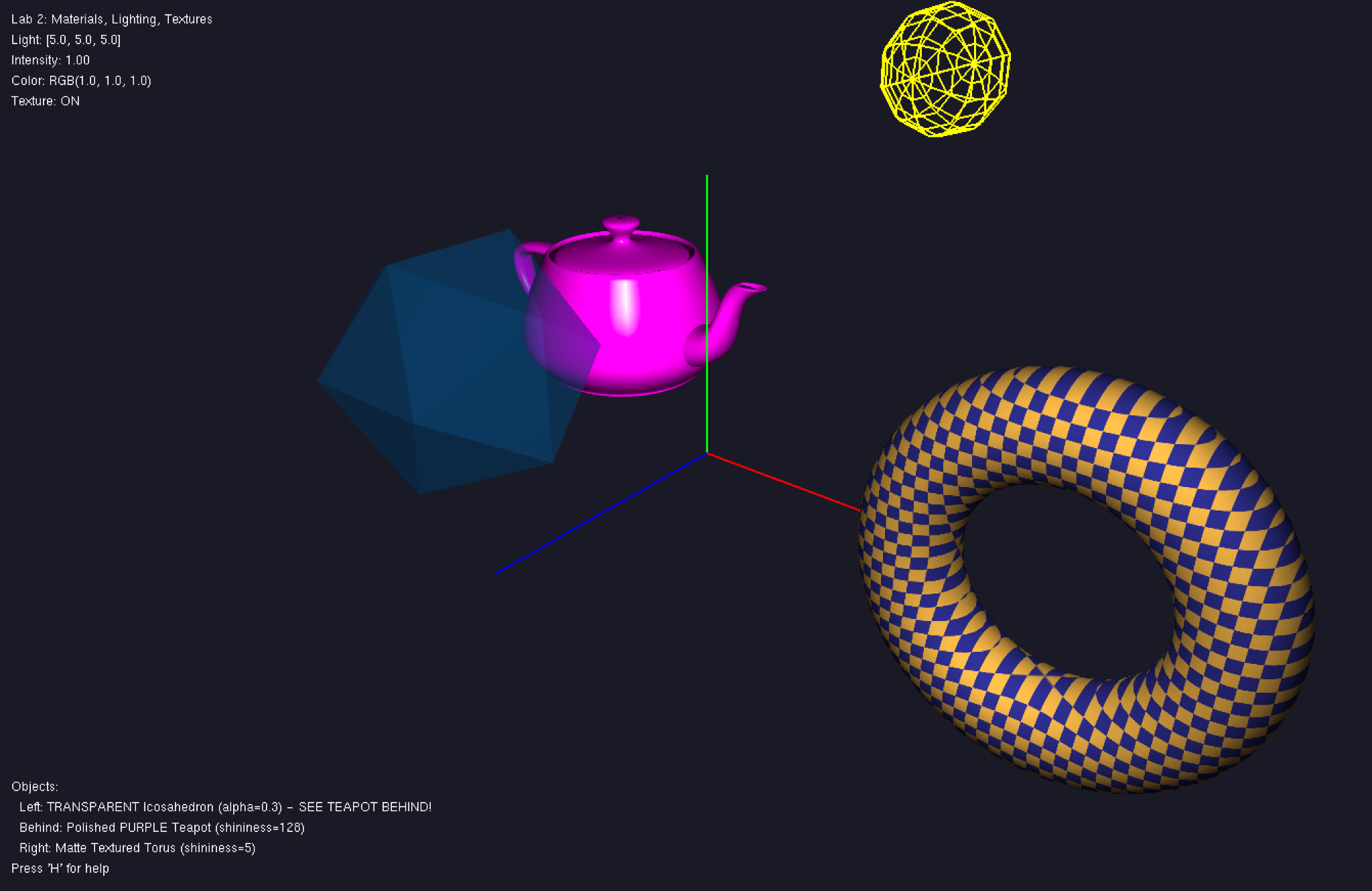
**

Рисунок 1. Отображение объектов сразу после запуска программы.

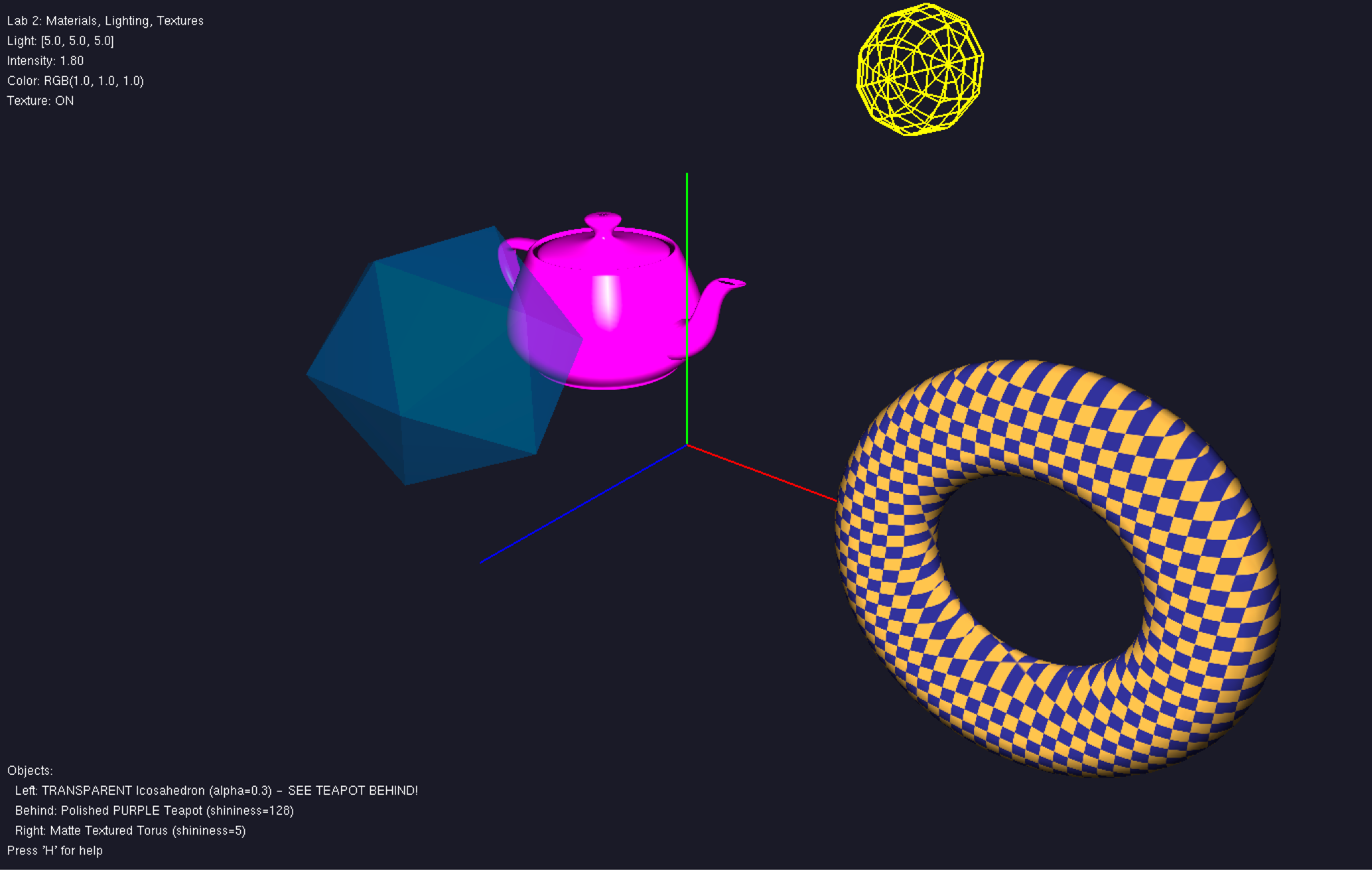
**

Рисунок 2. Отображение объектов после увеличения интенсивности освещения.

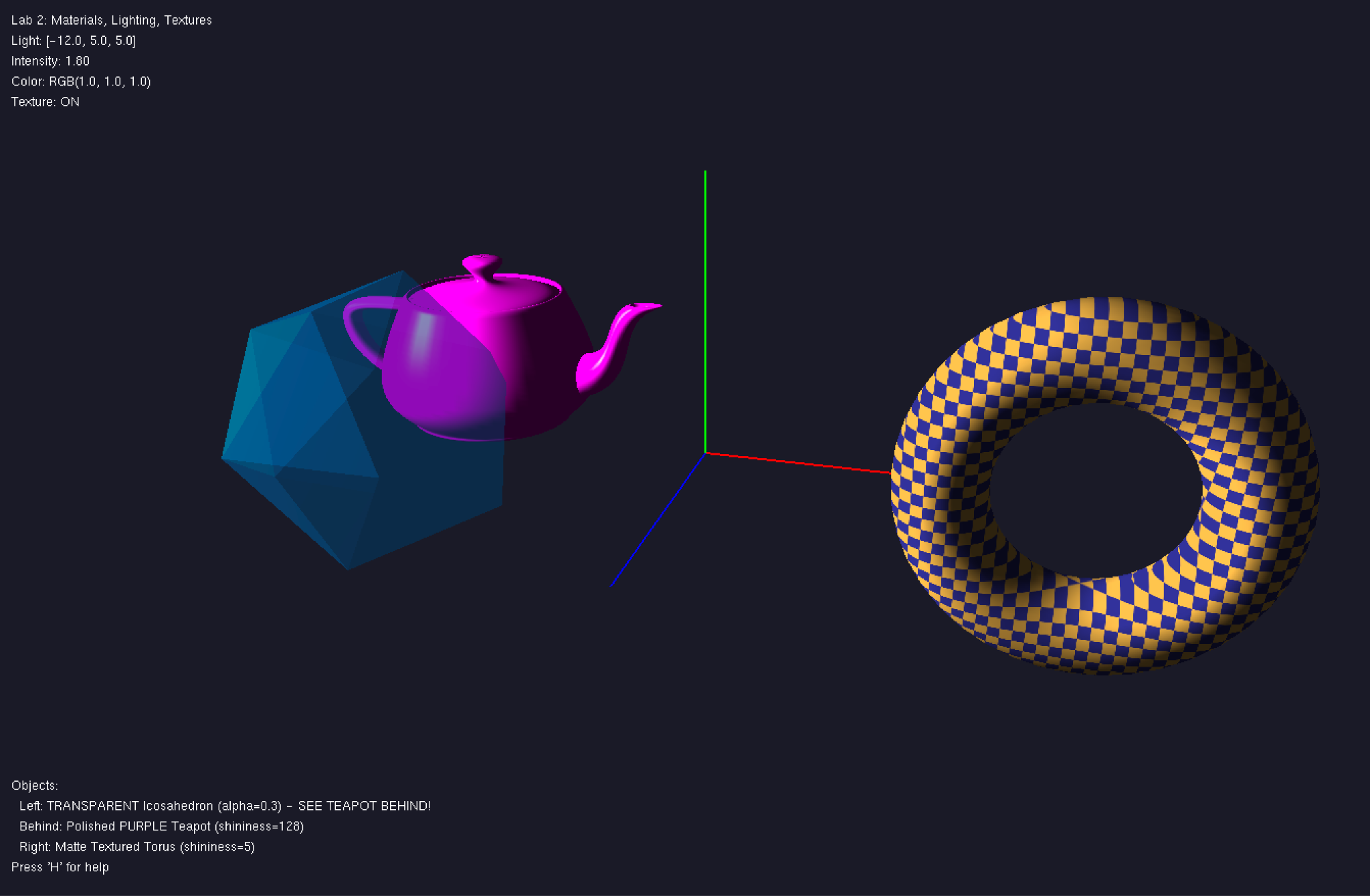
**

Рисунок 3. Отображение объектов после изменения местоположения источника освещения.

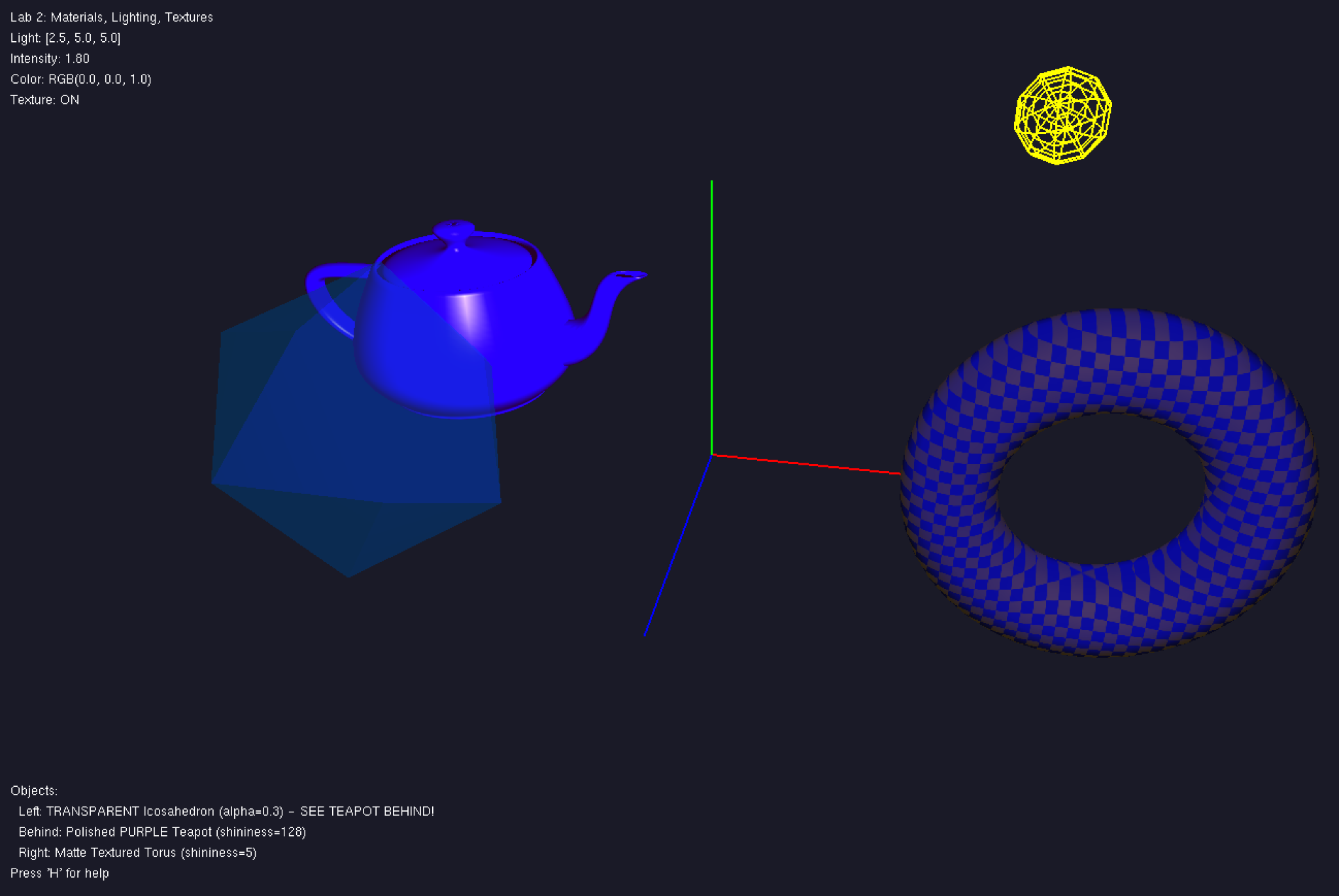
**

Рисунок 4. Отображение объектов после изменения цвета источника освещения (синий).

**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 = 1400 window\_height = 900  # Управление камерой mouse\_down = False mouse\_x = 0 mouse\_y = 0 rotation\_x = 30.0 rotation\_y = 45.0 zoom = 1.0  # Управление источником света light\_x = 5.0 light\_y = 5.0 light\_z = 5.0 light\_intensity = 1.0 light\_color = [1.0, 1.0, 1.0]  # Текстура texture\_id = None use\_texture = True   def create\_procedural\_texture():  """Создание процедурной текстуры (шахматная доска)"""  width, height = 256, 256  data = np.zeros((height, width, 3), dtype=np.uint8)   square\_size = 32  for i in range(height):  for j in range(width):  if ((i // square\_size) + (j // square\_size)) % 2 == 0:  data[i, j] = [255, 200, 100]  else:  data[i, j] = [50, 50, 150]   return data   def load\_texture():  """Загрузка текстуры"""  global texture\_id  texture\_data = create\_procedural\_texture()  texture\_id = glGenTextures(1)  glBindTexture(GL\_TEXTURE\_2D, texture\_id)  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT)  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT)  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR)  glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR)  glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, 256, 256, 0,  GL\_RGB, GL\_UNSIGNED\_BYTE, texture\_data)  glBindTexture(GL\_TEXTURE\_2D, 0)  print("✓ Текстура загружена")   def init\_opengl():  """Инициализация параметров OpenGL"""  glClearColor(0.1, 0.1, 0.15, 1.0)  glEnable(GL\_DEPTH\_TEST)   glMatrixMode(GL\_PROJECTION)  glLoadIdentity()  gluPerspective(45, window\_width / window\_height, 0.1, 50.0)  glMatrixMode(GL\_MODELVIEW)   glEnable(GL\_LIGHTING)  glEnable(GL\_LIGHT0)   # Прозрачность  glEnable(GL\_BLEND)  glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA)   load\_texture()  print("✓ OpenGL инициализирован")   def setup\_light():  """Настройка источника освещения"""  light\_position = [light\_x, light\_y, light\_z, 1.0]  glLightfv(GL\_LIGHT0, GL\_POSITION, light\_position)   ambient = [0.2 \* light\_intensity, 0.2 \* light\_intensity, 0.2 \* light\_intensity, 1.0]  diffuse = [light\_color[0] \* light\_intensity,  light\_color[1] \* light\_intensity,  light\_color[2] \* light\_intensity, 1.0]  specular = [1.0, 1.0, 1.0, 1.0]   glLightfv(GL\_LIGHT0, GL\_AMBIENT, ambient)  glLightfv(GL\_LIGHT0, GL\_DIFFUSE, diffuse)  glLightfv(GL\_LIGHT0, GL\_SPECULAR, specular)   def draw\_light\_marker():  """Отрисовка маркера источника света"""  glDisable(GL\_LIGHTING)  glDisable(GL\_TEXTURE\_2D)  glColor3f(1.0, 1.0, 0.0)  glPushMatrix()  glTranslatef(light\_x, light\_y, light\_z)  glutWireSphere(0.2, 10, 10)  glPopMatrix()  glEnable(GL\_LIGHTING)   def draw\_axes():  """Отрисовка осей координат"""  glDisable(GL\_LIGHTING)  glDisable(GL\_TEXTURE\_2D)  glLineWidth(2.0)   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()   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()   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()   glEnable(GL\_LIGHTING)   def draw\_transparent\_icosahedron():  """ОБЪЕКТ 1: Прозрачный икосаэдр (alpha = 0.3)"""  glDepthMask(GL\_FALSE)   # СИЛЬНО ПРОЗРАЧНЫЙ (alpha = 0.3)  ambient = [0.1, 0.4, 0.7, 0.3]  diffuse = [0.2, 0.6, 1.0, 0.3]  specular = [0.5, 0.5, 0.8, 0.3]  shininess = [30.0]   glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_AMBIENT, ambient)  glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE, diffuse)  glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SPECULAR, specular)  glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SHININESS, shininess)   glPushMatrix()  glTranslatef(-4.0, 0.0, 0.0)  glScalef(1.8, 1.8, 1.8)  glutSolidIcosahedron()  glPopMatrix()   glDepthMask(GL\_TRUE)   def draw\_polished\_teapot():  """ОБЪЕКТ 2: Отполированный чайник за икосаэдром"""  ambient = [0.25, 0.0, 0.25, 1.0]  diffuse = [1.0, 0.0, 1.0, 1.0]  specular = [1.0, 1.0, 1.0, 1.0]  shininess = [128.0]   glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_AMBIENT, ambient)  glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE, diffuse)  glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SPECULAR, specular)  glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SHININESS, shininess)   glPushMatrix()  glTranslatef(-4.0, 0.0, -3.0) # За икосаэдром  glutSolidTeapot(1.5)  glPopMatrix()   def draw\_textured\_torus():  """ОБЪЕКТ 3: Матовый тор с текстурой"""  ambient = [0.3, 0.3, 0.3, 1.0]  diffuse = [0.8, 0.8, 0.8, 1.0]  specular = [0.1, 0.1, 0.1, 1.0]  shininess = [5.0]   glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_AMBIENT, ambient)  glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE, diffuse)  glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SPECULAR, specular)  glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SHININESS, shininess)   if use\_texture and texture\_id:  glEnable(GL\_TEXTURE\_2D)  glBindTexture(GL\_TEXTURE\_2D, texture\_id)  glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE)  glEnable(GL\_TEXTURE\_GEN\_S)  glEnable(GL\_TEXTURE\_GEN\_T)  glTexGeni(GL\_S, GL\_TEXTURE\_GEN\_MODE, GL\_OBJECT\_LINEAR)  glTexGeni(GL\_T, GL\_TEXTURE\_GEN\_MODE, GL\_OBJECT\_LINEAR)   glPushMatrix()  glTranslatef(4.0, 0.0, 0.0)  glutSolidTorus(0.5, 1.5, 30, 40)  glPopMatrix()   if use\_texture:  glDisable(GL\_TEXTURE\_GEN\_S)  glDisable(GL\_TEXTURE\_GEN\_T)  glDisable(GL\_TEXTURE\_2D)   def draw\_text(x, y, text):  """Отрисовка текста на экране"""  glDisable(GL\_DEPTH\_TEST)  glDisable(GL\_LIGHTING)  glDisable(GL\_TEXTURE\_2D)   glMatrixMode(GL\_PROJECTION)  glPushMatrix()  glLoadIdentity()  gluOrtho2D(0, window\_width, 0, window\_height)   glMatrixMode(GL\_MODELVIEW)  glPushMatrix()  glLoadIdentity()   glColor3f(1.0, 1.0, 1.0)  glRasterPos2f(x, y)  for char in text:  glutBitmapCharacter(GLUT\_BITMAP\_HELVETICA\_12, ord(char))   glPopMatrix()  glMatrixMode(GL\_PROJECTION)  glPopMatrix()  glMatrixMode(GL\_MODELVIEW)   glEnable(GL\_DEPTH\_TEST)  glEnable(GL\_LIGHTING)   def apply\_camera\_rotation():  """Применение вращения камеры"""  distance = 12.0 \* 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():  """Основная функция отображения"""  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)   setup\_light()  draw\_axes()  draw\_light\_marker()   # ПОРЯДОК: непрозрачные → прозрачные  draw\_textured\_torus()  draw\_polished\_teapot() # За икосаэдром  draw\_transparent\_icosahedron() # Прозрачный последним!   draw\_text(20, window\_height - 30, "Lab 2: Materials, Lighting, Textures")  draw\_text(20, window\_height - 50, f"Light: [{light\_x:.1f}, {light\_y:.1f}, {light\_z:.1f}]")  draw\_text(20, window\_height - 70, f"Intensity: {light\_intensity:.2f}")  draw\_text(20, window\_height - 90, f"Color: RGB({light\_color[0]:.1f}, {light\_color[1]:.1f}, {light\_color[2]:.1f})")  draw\_text(20, window\_height - 110, f"Texture: {'ON' if use\_texture else 'OFF'}")  draw\_text(20, 120, "Objects:")  draw\_text(20, 100, " Left: TRANSPARENT Icosahedron (alpha=0.3) - SEE TEAPOT BEHIND!")  draw\_text(20, 80, " Behind: Polished PURPLE Teapot (shininess=128)")  draw\_text(20, 60, " Right: Matte Textured Torus (shininess=5)")  draw\_text(20, 40, "Press 'H' for help")   glutSwapBuffers()   def keyboard(key, x, y):  """Обработка нажатий клавиш"""  global rotation\_x, rotation\_y, zoom, light\_intensity, light\_color, use\_texture  global light\_x, light\_y, light\_z   if key == b'r' or key == b'R':  rotation\_x = 30.0  rotation\_y = 45.0  zoom = 1.0  print("Камера сброшена")  elif key == b'+' or key == b'=':  light\_intensity = min(2.0, light\_intensity + 0.1)  print(f"Интенсивность: {light\_intensity:.2f}")  elif key == b'-' or key == b'\_':  light\_intensity = max(0.0, light\_intensity - 0.1)  print(f"Интенсивность: {light\_intensity:.2f}")  elif key == b'w' or key == b'W':  light\_y += 0.5  print(f"Свет Y={light\_y:.1f}")  elif key == b's' or key == b'S':  light\_y -= 0.5  print(f"Свет Y={light\_y:.1f}")  elif key == b'a' or key == b'A':  light\_x -= 0.5  print(f"Свет X={light\_x:.1f}")  elif key == b'd' or key == b'D':  light\_x += 0.5  print(f"Свет X={light\_x:.1f}")  elif key == b'q' or key == b'Q':  light\_z += 0.5  print(f"Свет Z={light\_z:.1f}")  elif key == b'e' or key == b'E':  light\_z -= 0.5  print(f"Свет Z={light\_z:.1f}")  elif key == b'1':  light\_color = [1.0, 0.0, 0.0]  print("Свет: Красный")  elif key == b'2':  light\_color = [0.0, 1.0, 0.0]  print("Свет: Зелёный")  elif key == b'3':  light\_color = [0.0, 0.0, 1.0]  print("Свет: Синий")  elif key == b'4':  light\_color = [1.0, 1.0, 0.0]  print("Свет: Жёлтый")  elif key == b'5':  light\_color = [1.0, 1.0, 1.0]  print("Свет: Белый")  elif key == b't' or key == b'T':  use\_texture = not use\_texture  print(f"Текстура: {'ВКЛ' if use\_texture else 'ВЫКЛ'}")  elif key == b'h' or key == b'H':  print("\n" + "=" \* 80)  print("УПРАВЛЕНИЕ")  print("=" \* 80)  print("Камера: ЛКМ+движение, колесико, R")  print("Свет: W/A/S/D/Q/E (позиция), +/- (интенсивность), 1-5 (цвет)")  print("Текстура: T")  print("=" \* 80 + "\n")  elif key == b'\x1b':  sys.exit(0)   glutPostRedisplay()   def mouse(button, state, x, y):  """Обработка 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):  """Движение мыши"""  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  rotation\_x = max(-89.0, min(89.0, rotation\_x))  mouse\_x = x  mouse\_y = y  glutPostRedisplay()   def main():  glutInit(sys.argv)  glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB | GLUT\_DEPTH | GLUT\_ALPHA)  glutInitWindowSize(window\_width, window\_height)  glutInitWindowPosition(100, 100)  glutCreateWindow(b"Lab 2: Materials, Lighting, Textures")   init\_opengl()  glutDisplayFunc(display)  glutKeyboardFunc(keyboard)  glutMouseFunc(mouse)  glutMotionFunc(motion)   print("=" \* 80)  print("Задание 2: Материалы, Освещение, Текстуры")  print("=" \* 80)  print("\n✓ Вариант #10: икосаэдр, конус, чайник, тор")  print("\n✓ ОБЪЕКТЫ:")  print(" 1. Икосаэдр (слева) - ПРОЗРАЧНЫЙ (alpha=0.3)")  print(" 2. Чайник (за ним) - Отполированный (shininess=128)")  print(" 3. Тор (справа) - Матовый текстурированный")  print("\n✓ Освещение: точечный источник (W/A/S/D/Q/E, +/-, 1-5)")  print("✓ Текстура: процедурная шахматная доска (T)")  print("=" \* 80)   glutMainLoop()   if \_\_name\_\_ == "\_\_main\_\_":  main() |
| --- |