Баганов В.С.

2 вариант

Группа в3530904/00330

# Задание

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 от 0,9 до 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. Возможно при этом также использовать микроискажение нормалей при помощи bump-mapping

# Выполнение

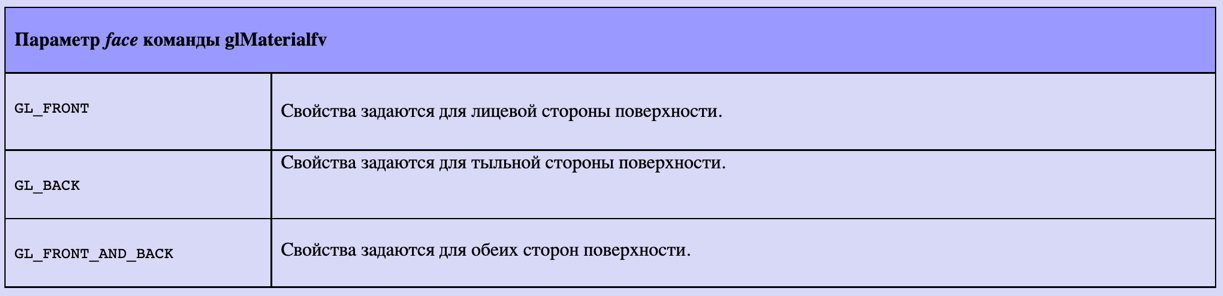
Используемая среда разработки Pycharm с языком Python.Для реализации была выбрана библиотека PyOpenGL и GLUT.

В начале программы запускаются функции: glutInit(), rgbglutInitDisplayMode(), glutInitWindowSize(),glutCreateWindow(). В них задаются параметры окна.

Дальше идёт функция glutDisplayFunc(), она ответственна за вывод на экран, в неё необходимо передать функцию которая будет отрисовывать объекты. В ней последовательно отрисовывается куб, конус и сфера.

Объекты задаются функциями glutWireTeapot(), glutWireOctahedron(), gluCylinder(). Для того что бы их менять существуют следующие функции:

1. glColor4f – в функции задаётся цвет
2. glRotatef – эта функция отвечает за поворот
3. glTranslated – тут определяется точка в которой будет отрисовывается объект
4. glScalef – а эта функция отвечает за увеличение объекта
5. glMaterialfv – свойства материала отражать различные компоненты освещения: фоновую, рассеянную и зеркальную





Функции glutVisibilityFunc() и glutReshapeFunc() нужны что бы перерисовывать объекты после изменений, а функция glutMainLoop() запускает цикл для того, что бы перерисовка происходила постоянно.

В начале программы инициализируется окно с 3 примитивами. Конус прозрачный, куб текстурированный, а сфера полированная.

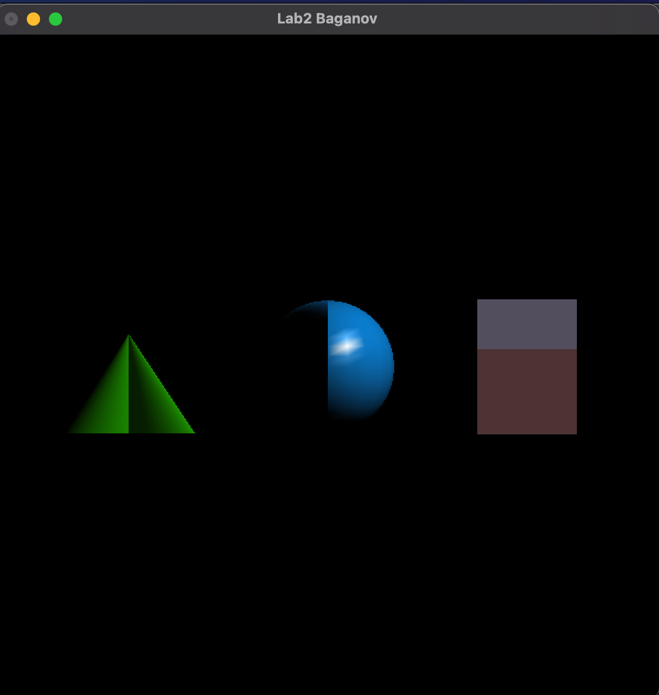
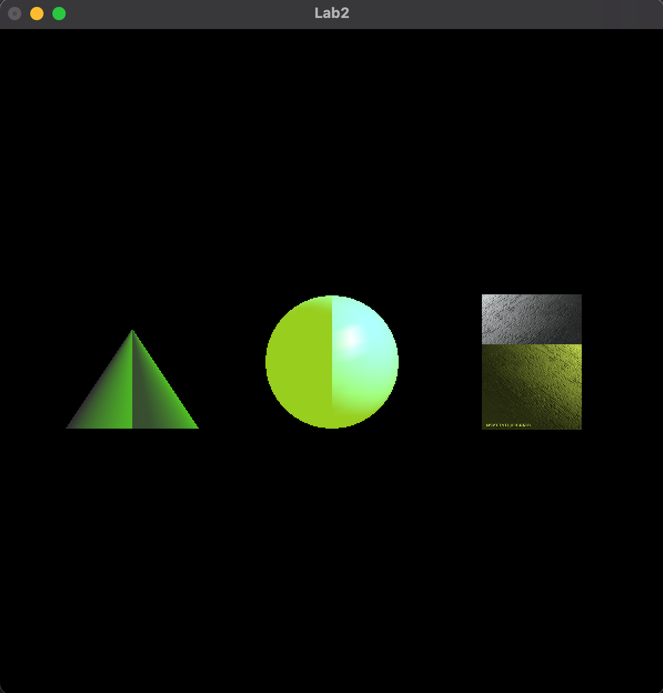


Рисунок 1. Инициализация программы

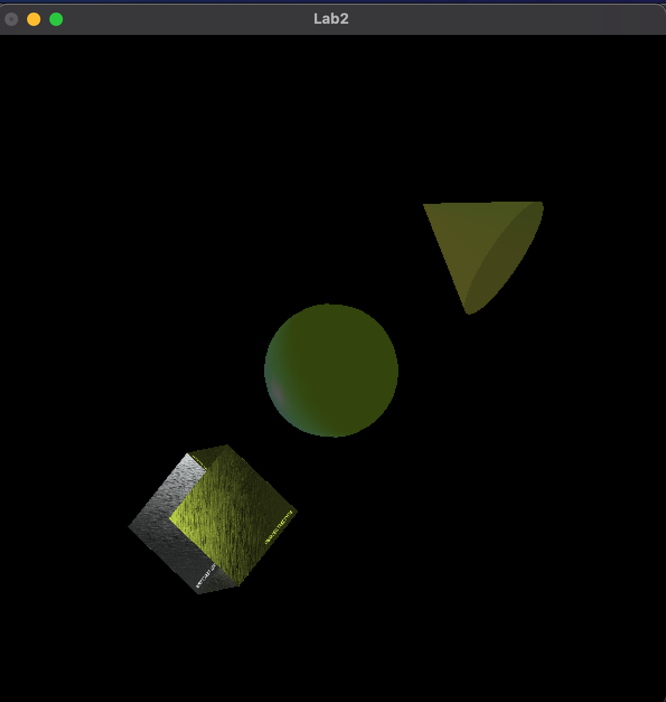
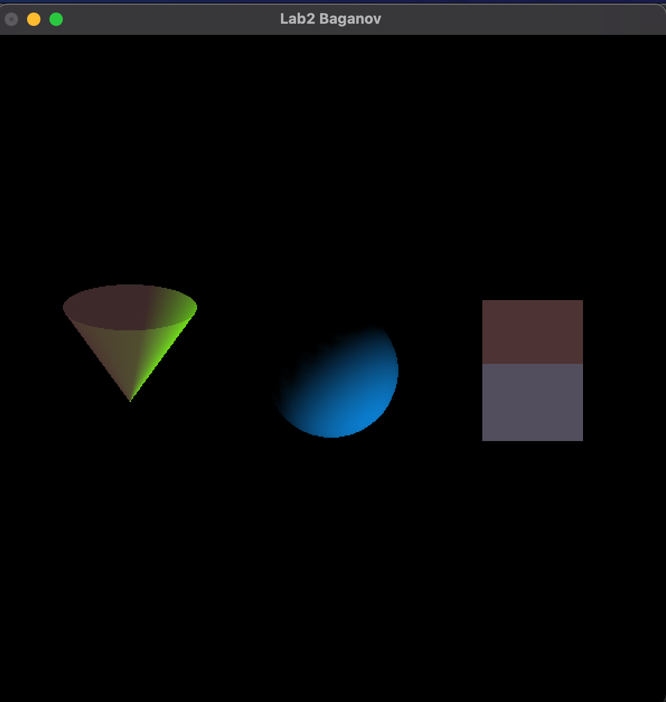


Рисунок 2. Пример поворота

С помощью кнопки F6 можно сменить цвет освещения.

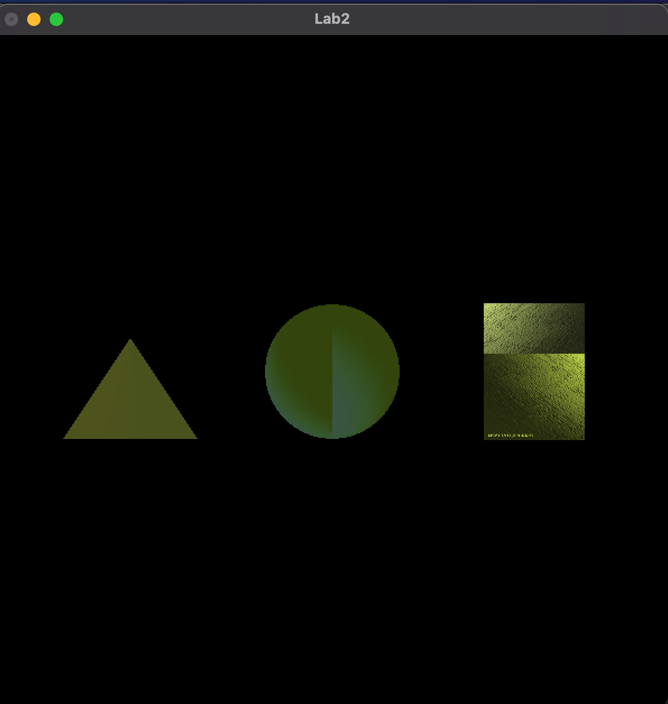
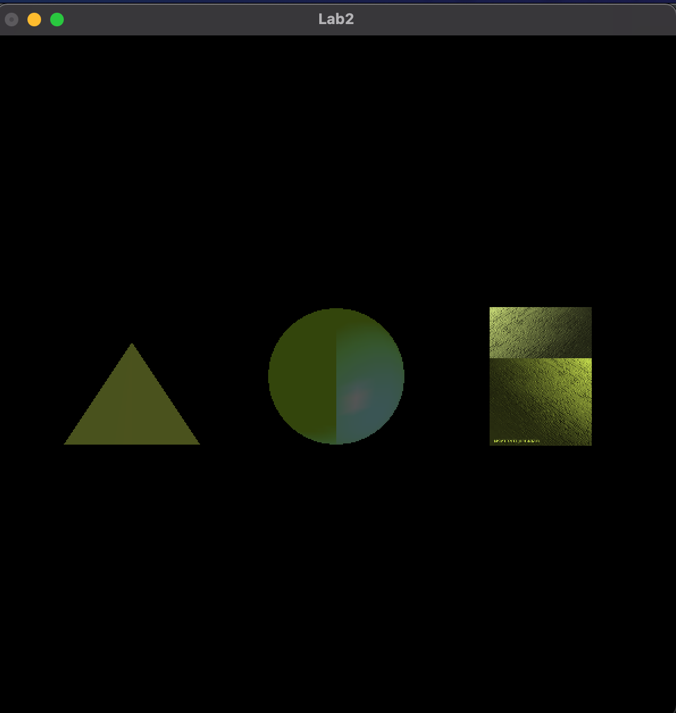
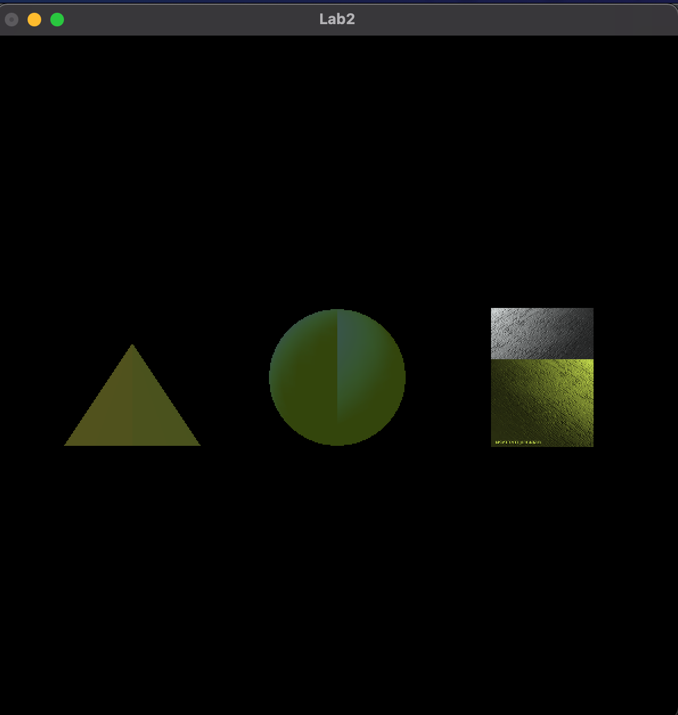


Рисунок 3. Изменение направления света

С помощью кнопок F2 и F3 можно сменить направление освещения.

# Приложение

from OpenGL.GL import \*  
from OpenGL.GLU import \*  
from OpenGL.GLUT import \*  
  
import sys  
from PIL import Image  
from numpy import array, int8  
  
# Объявляем все глобальные переменные  
global xrot # Величина вращения по оси x  
global yrot # Величина вращения по оси y  
global zrot  
global cube\_z\_coor  
global ambient # рассеянное освещение  
global greencolor # Цвет елочных иголок  
global treecolor # Цвет елочного стебля  
global lightpos # Положение источника освещения  
global light\_ambient  
global light\_diffuse  
global light\_specular  
global light\_position  
  
# Процедура инициализации  
def init():  
 global xrot # Величина вращения по оси x  
 global yrot # Величина вращения по оси y  
 global zrot  
 global cube\_z\_coor  
 global ambient # Рассеянное освещение  
 global greencolor # Цвет елочных иголок  
 global treecolor # Цвет елочного ствола  
 global lightpos # Положение источника освещения  
 global cube\_texture  
 global cube\_display\_list  
  
 xrot = 0.0 # Величина вращения по оси x = 0  
 yrot = 0.0 # Величина вращения по оси y = 0  
 zrot = 0.0  
 cube\_z\_coor=0.0  
 # ambient = (.6, .6, .6, 1) # Первые три числа цвет в формате RGB, а последнее - яркость  
 # lightpos = (1.0, 1.0, 1.0)  
 ambient = (1., 1., 1.0, 1) # Первые три числа цвет в формате RGB, а последнее - яркость  
 greencolor = (0.2, 0.8, 0.0, 0.8) # Зеленый цвет для иголок  
 treecolor = (0.9, 0.6, 0.3, 0.1) # Коричневый цвет для ствола  
 lightpos = (1.0, 1.0, 1.0) # Положение источника освещения по осям xyz  
 light\_ambient = (.2, .2, .2, 1)  
 light\_diffuse = array([1, 1, 1, 1])  
 light\_specular = (1, 1, 1, 1)  
 light\_position = [0, 1, .5, 0]  
  
 glClearColor(0.0, 0.0, 0.0, 1.0) # Серый цвет для первоначальной закраски  
 gluOrtho2D(-1.0, 1.0, -1.0, 1.0) # Определяем границы рисования по горизонтали и вертикали  
 glRotatef(-90, 1.0, 0.0, 0.0) # Сместимся по оси Х на 90 градусов  
 glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, ambient) # Определяем текущую модель освещения  
 glEnable(GL\_LIGHTING) # Включаем освещение  
 glEnable(GL\_LIGHT0) # Включаем один источник света  
 glLightfv(GL\_LIGHT0, GL\_POSITION, lightpos) # Определяем положение источника света  
  
  
  
def draw\_cube(s): # задаем вершины куба и точки натяжения текстуры  
 s /= 2  
  
 glBegin(GL\_QUADS)  
  
 # face  
 glNormal(0, 0, 1)  
 glTexCoord(0, 1)  
 glVertex(-s, -s, s)  
 glTexCoord(1, 1)  
 glVertex(s, -s, s)  
 glTexCoord(1, 0)  
 glVertex(s, s, s)  
 glTexCoord(0, 0)  
 glVertex(-s, s, s)  
  
 # back  
 glNormal(0, 0, -1)  
 glTexCoord(1, 1)  
 glVertex(s, -s, -s)  
 glTexCoord(1, 0)  
 glVertex(s, s, -s)  
 glTexCoord(0, 0)  
 glVertex(-s, s, -s)  
 glTexCoord(0, 1)  
 glVertex(-s, -s, -s)  
  
 # left  
 glNormal(-1, 0, 0)  
 glTexCoord(0, 0)  
 glVertex(-s, -s, -s)  
 glTexCoord(1, 0)  
 glVertex(-s, -s, s)  
 glTexCoord(1, 1)  
 glVertex(-s, s, s)  
 glTexCoord(0, 1)  
 glVertex(-s, s, -s)  
  
 # right  
 glNormal(1, 0, 0)  
 glTexCoord(0, 0)  
 glVertex(s, -s, s)  
 glTexCoord(0, 1)  
 glVertex(s, s, s)  
 glTexCoord(1, 1)  
 glVertex(s, s, -s)  
 glTexCoord(1, 0)  
 glVertex(s, -s, -s)  
  
 # top  
 glNormal(0, 1, 0)  
 glTexCoord(0, 0)  
 glVertex(-s, s, s)  
 glTexCoord(1, 0)  
 glVertex(s, s, s)  
 glTexCoord(1, 1)  
 glVertex(s, s, -s)  
 glTexCoord(0, 1)  
 glVertex(-s, s, -s)  
  
 # bottom  
 glNormal(0, -1, 0)  
 glTexCoord(0, 0)  
 glVertex(s, -s, s)  
 glTexCoord(0, 1)  
 glVertex(s, -s, -s)  
 glTexCoord(1, 1)  
 glVertex(-s, -s, -s)  
 glTexCoord(1, 0)  
 glVertex(-s, -s, s)  
  
 glEnd()  
  
def load\_texture(filename):  
 img = Image.open(filename)  
 img\_data = array(list(img.getdata()), int8)  
  
 texture\_id = glGenTextures(1)  
 glBindTexture(GL\_TEXTURE\_2D, texture\_id)  
 glPixelStorei(GL\_UNPACK\_ALIGNMENT, 1)  
 #  
 glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_S, GL\_REPEAT)  
 glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_WRAP\_T, GL\_REPEAT)  
 glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MAG\_FILTER, GL\_LINEAR)  
 glTexParameterf(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR)  
 glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, img.size[0], img.size[1], 0, GL\_RGB, GL\_UNSIGNED\_BYTE, img\_data)  
  
 glBindTexture(GL\_TEXTURE\_2D, 0)  
  
 return texture\_id  
  
# Процедура обработки специальных клавиш  
def specialkeys(key, x, y):  
 global xrot  
 global yrot  
 global zrot  
 global cube\_z\_coor  
 global light\_position  
 global light\_diffuse  
 global lin\_att  
 global quad\_att  
 global lightpos  
  
 # Обработчики для клавиш со стрелками  
 if key == GLUT\_KEY\_UP: # Клавиша вверх  
 xrot -= 2.0 # Уменьшаем угол вращения по оси Х  
 if key == GLUT\_KEY\_DOWN: # Клавиша вниз  
 xrot += 2.0 # Увеличиваем угол вращения по оси Х  
 if key == GLUT\_KEY\_LEFT: # Клавиша влево  
 yrot -= 2.0 # Уменьшаем угол вращения по оси Y  
 if key == GLUT\_KEY\_RIGHT: # Клавиша вправо  
 yrot += 2.0 # Увеличиваем угол вращения по оси Y  
 if key == GLUT\_KEY\_F7:  
 zrot -= 10.0  
 if key == GLUT\_KEY\_F8:  
 zrot += 10.0  
 if key == GLUT\_KEY\_F9:  
 cube\_z\_coor += 0.01  
 if key == GLUT\_KEY\_F10:  
 cube\_z\_coor -= 0.01  
 if key == GLUT\_KEY\_F12:  
 # zrot -= 10.0  
 exit(0)  
 if key == GLUT\_KEY\_F1:  
 lightpos = (-1.0, 1.0, 1.0)  
 if key == GLUT\_KEY\_F2:  
 lightpos = (-1.0, 1.0, -1.0)  
 if key == GLUT\_KEY\_F3:  
 lightpos = (1.0, 1.0, -1.0)  
 if key == GLUT\_KEY\_F4:  
 lin\_att += 0.1  
 quad\_att += 0.2  
 if key == GLUT\_KEY\_F5:  
 if lin\_att <= 0:  
 lin\_att = 0  
 quad\_att = 0  
 else:  
 lin\_att -= 0.1  
 quad\_att -= 0.2  
  
 glutPostRedisplay() # Вызываем процедуру перерисовки  
  
  
# Процедура перерисовки  
def draw():  
 global xrot  
 global yrot  
 global lightpos  
 global greencolor  
 global treecolor  
 global cube\_z  
 global cube\_texture  
  
 glClear(GL\_COLOR\_BUFFER\_BIT) # Очищаем экран и заливаем серым цветом  
 glPushMatrix() # Сохраняем текущее положение "камеры"  
 glRotatef(xrot, 1.0, 0.0, 0.0) # Вращаем по оси X на величину xrot  
 glRotatef(yrot, 0.0, 1.0, 0.0) # Вращаем по оси Y на величину yrot  
 glRotate(zrot, 0.0, 0.0, 1.0)  
  
 glLightfv(GL\_LIGHT0, GL\_POSITION, lightpos) # Источник света вращаем вместе   
 glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE, greencolor)  
  
 # сдвигаемся влево и вниз, чтобы отрисовать  
 glTranslatef(-0.6, 0.0, -0.2)  
 glBlendFunc(GL\_SRC\_ALPHA, GL\_ONE\_MINUS\_SRC\_ALPHA)  
 glEnable(GL\_BLEND)  
  
  
 # glutSolidCylinder(0.2, 0.75, 200, 200)  
 glutSolidCone(0.2, 0.3, 100, 100)  
  
 glDisable(GL\_BLEND)  
  
 #снова возвращаем систему координат в начало и отрисовывавем сферу  
 no\_mat = [0.0, 0.0, 0.0, 1.0];  
 mat\_ambient = [0.7, 0.7, 0.7, 1.0];  
 mat\_ambient\_color = [0.6, 0.8, 0.2, 1.0];  
 mat\_diffuse = [0.1, 0.5, 0.8, 1.0];  
 mat\_specular = [1.0, 1.0, 1.0, 1.0]  
 no\_shininess = 0.0;  
 low\_shininess = 5.0;  
 high\_shininess = 128.0;  
 mat\_emission = [0.3, 0.2, 0.2, 0.0];  
  
 glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat\_ambient\_color);  
 glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);  
 glMaterialfv(GL\_FRONT, GL\_SPECULAR, mat\_specular);  
 glMaterialf(GL\_FRONT, GL\_SHININESS, high\_shininess);  
 glMaterialfv(GL\_FRONT, GL\_EMISSION, no\_mat);  
  
 #glMaterialf(GL\_FRONT, GL\_SHININESS, 0.0);  
 glTranslatef(0.6, 0.0, 0.2)  
 glutSolidSphere(0.2, 20, 20)  
  
 #сдвигаемся вправо и рисуем куб или чайник  
 glMaterialfv(GL\_FRONT, GL\_AMBIENT, mat\_ambient\_color);  
 # glMaterialfv(GL\_FRONT, GL\_AMBIENT, no\_mat);  
 glMaterialfv(GL\_FRONT, GL\_DIFFUSE, mat\_diffuse);  
 glMaterialfv(GL\_FRONT, GL\_SPECULAR, no\_mat);  
 # glMaterialf(GL\_FRONT, GL\_SHININESS, no\_shininess);  
 glMaterialf(GL\_FRONT, GL\_SHININESS, low\_shininess);  
 glMaterialfv(GL\_FRONT, GL\_EMISSION, mat\_emission);  
 # glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, light\_ambient)  
  
  
 glTranslatef(0.6, 0.0, 0.00)  
 glRotatef(0, 1.0, 0.0, 0.0)  
  
 cube\_texture = load\_texture('test1.jpeg')  
 # cube\_texture = load\_texture('yachik3.jpeg')  
 glBindTexture(GL\_TEXTURE\_2D, cube\_texture)  
 cube\_display\_list = glGenLists(1)  
 glNewList(cube\_display\_list, GL\_COMPILE)  
 # glutSolidCube(0.3)  
 # glutSolidTeapot(0.2)  
  
 glRotatef(30, 1.0, 0.0, 0.0)  
 draw\_cube(0.3)  
 # draw\_tetrahedron(1.1)  
 glEndList()  
 glEnable(GL\_TEXTURE\_2D)  
 glBindTexture(GL\_TEXTURE\_2D, cube\_texture)  
 glCallList(cube\_display\_list)  
  
  
 # glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, light\_ambient)  
 glEnable(GL\_LIGHTING)  
  
  
 glPopMatrix() # Возвращаем сохраненное положение "камеры"  
 glutSwapBuffers() # Выводим все нарисованное в памяти на экран  
  
  
# execution starts here  
# set the initial display mode  
# use double buffering and rgb colors  
glutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB)  
  
# set the initial window size - width and height  
glutInitWindowSize(600, 600)  
  
# определяем положение окна с фигурами  
glutInitWindowPosition(400, 50)  
  
# initialize opengl  
glutInit(sys.argv)  
  
# create the main window  
glutCreateWindow("Lab2")  
  
# redraw  
glutDisplayFunc(draw)  
  
# this procedure handles keystrokes  
glutSpecialFunc(specialkeys)  
  
# call the initialization function  
init()  
  
# run the main loop  
glutMainLoop()