Баганов В.C.

2 вариант

Группа в3530904/00330

# Задание №1

1. Изобразить чайник и цилиндр. Размеры и местоположение примитивов на экране задать самостоятельно.
2. Переместить чайник так, чтобы он был расположен на цилиндре.
3. Изобразить конус и цилиндр, одинаковой высоты, центры основания конуса и цилиндра совпадают.
4. Промасштабировать конус с коэффициентом 1.25, переместить цилиндр на dх=30

# 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.

# инициализация программы

# устанавливаем начальный режим отображения

# используем двойную буферизацию и цвета

rgbglutInitDisplayMode(GLUT\_DOUBLE | GLUT\_RGB)

# устанавливаем начальный размер окна - ширину и высоту

glutInitWindowSize(600, 600)

# оцениваем положение окна с фигурами

glutInitWindowPosition(900, 200 )

# инициализировать

openglglutInit(sys.argv)

# создать главное окно

glutCreateWindow("Lab1\_Baganov")

redrawglutDisplayFunc(draw)

# эта процедура обрабатывает нажатия клавиш

glutSpecialFunc(specialkeys)

glutDisplayFunc(renderScene);

glutReshapeFunc (изменить форму);

glutIdleFunc(renderScene);

# glutSpecialFunc(control\_proc);

# вызвать функцию инициализации init()# запустить основной цикл

glutMainLoop()

Дальше идёт функция glutDisplayFunc(), она ответственна за вывод на экран, в неё необходимо передать функцию которая будет отрисовывать объекты. В нашем случае это функция display\_3d\_2l. В ней последовательно отрисовывается чайник, цилиндр, конус, куб и цилиндр.

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

1. glColor3f – 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

Функция glutReshapeFunc() нужна что бы перерисовывать объекты после изменений, а функция glutMainLoop() запускает цикл для того, что бы перерисовка происходила постоянно.

В начале программы инициализируется окно с 6 примитивами для чтобы научиться строили различные примитивы.

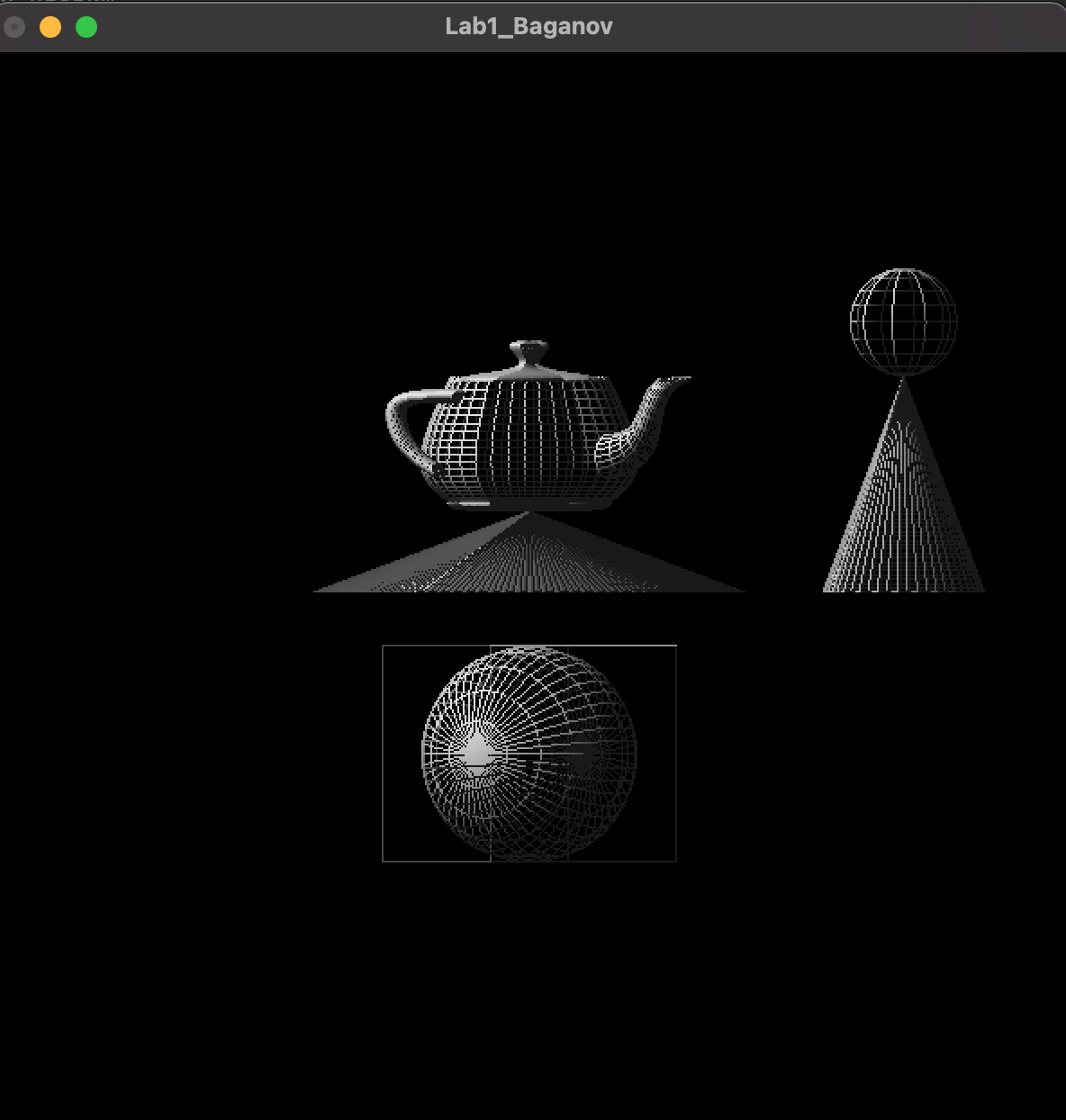


Рисунок 1. Пример использования различных примитивов

Управление осуществляется стрелочками клавиатуры. Перемещение чайника на цилиндр осуществляется по нажатию стрелочек вправо и влево.

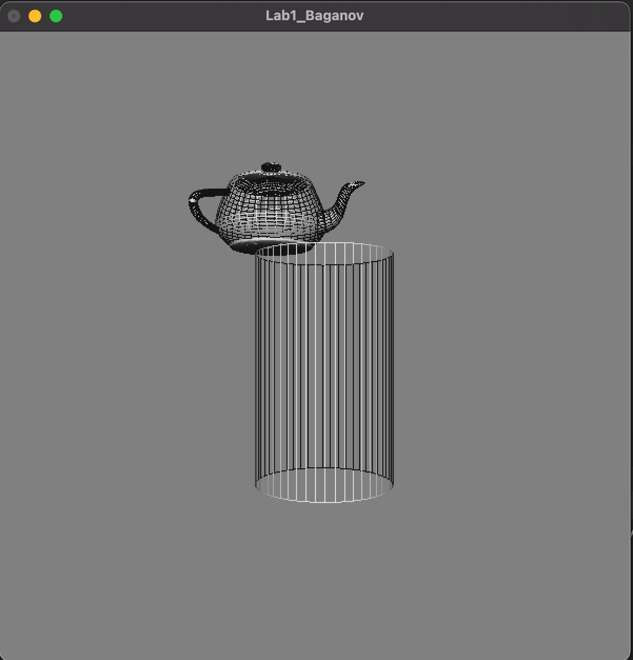


Рисунок 2. Пример перемещения чайника на цилиндр

Управление осуществляется стрелочками клавиатуры. Перемещение чайника на цилиндр осуществляется по нажатию стрелочек вправо и влево.

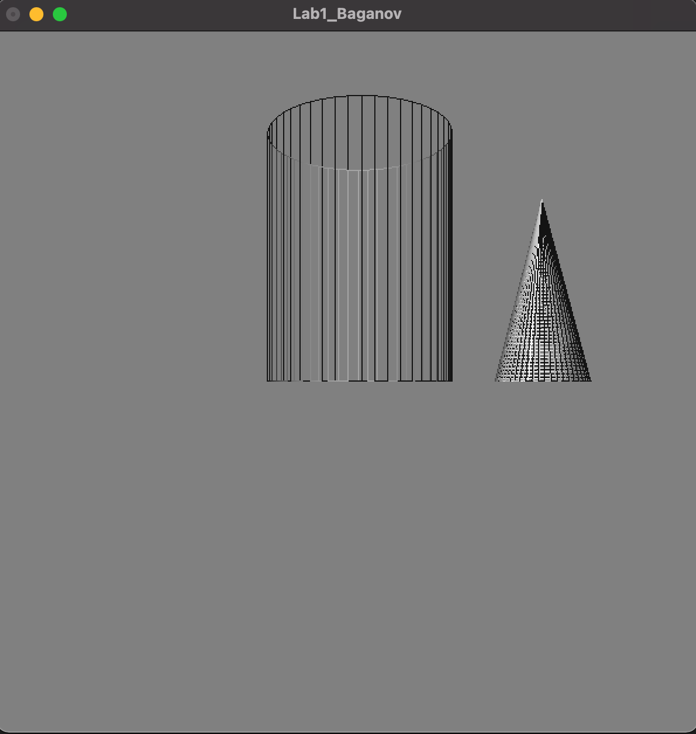
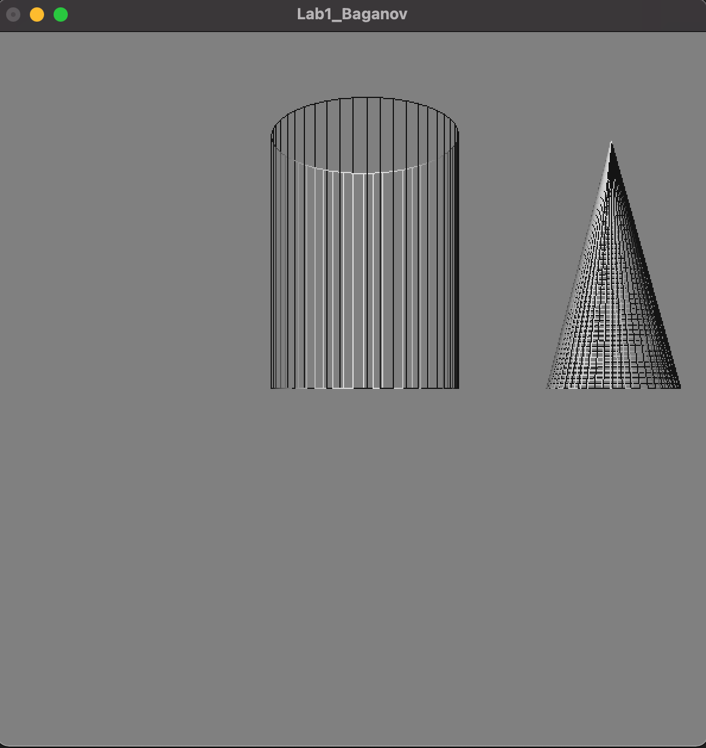
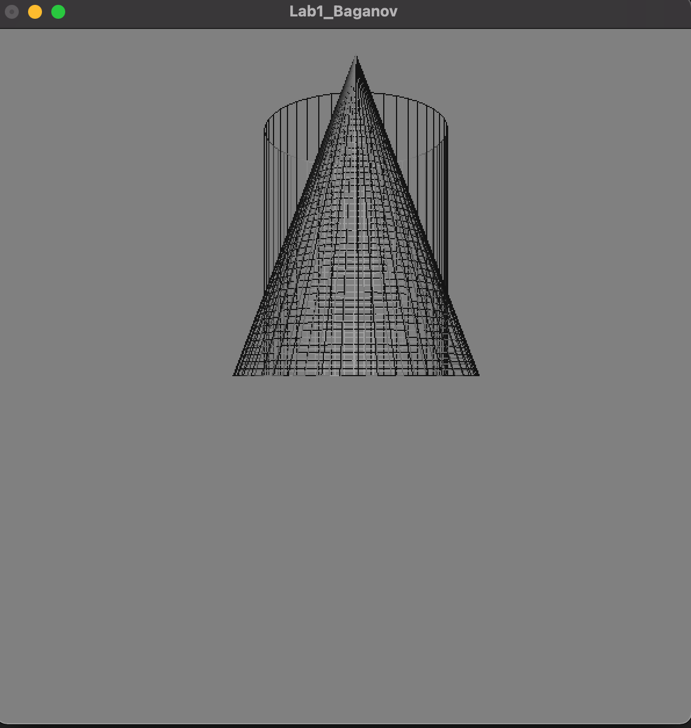
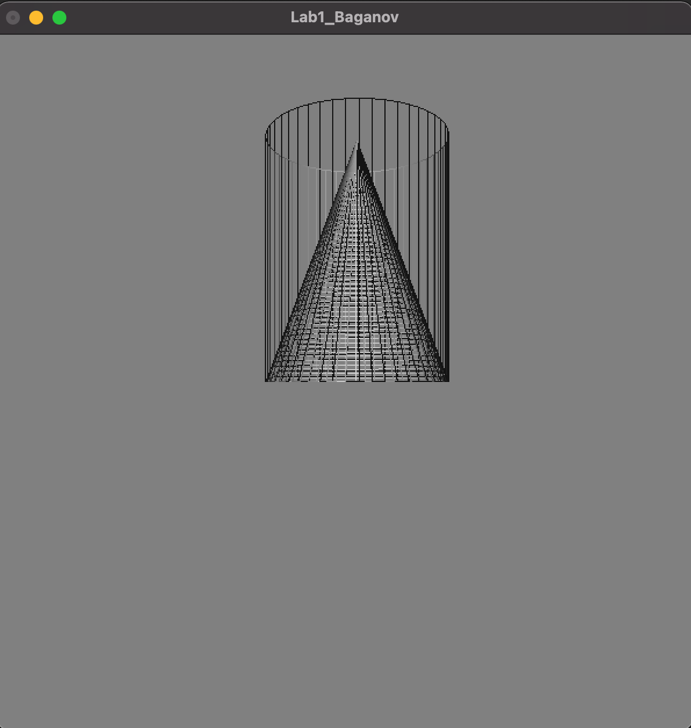


Рисунок 3. Масштабирование и перемещение

Для увеличения масштаба и перемещения фигуры используются стрелочки вверх и вниз.

# Приложение

# Импортируем все необходимые библиотеки:  
from OpenGL.GL import \*  
from OpenGL.GLU import \*  
from OpenGL.GLUT import \*  
import sys  
import random  
  
  
  
# Объявляем все глобальные переменные  
global xrot  
global yrot  
global zrot  
global cube\_z\_coor  
global ambient  
global lightpos  
  
  
# float scale\_tetra\_Z = 5.0;  
# float scale\_tetra\_X = 5.0;  
# float scale\_tetra\_Mod = 0.75;  
# int pot\_pos = -6;  
# int sphere\_pos = 0;  
# int rot = 0;  
  
# Процедура инициализации  
def init():  
 global scale\_tetra\_Y  
 global scale\_tetra\_Z  
 global scale\_tetra\_X  
 global scale\_tetra\_Mod  
 global pot\_pos  
 global sphere\_pos  
 global rot  
 global xrot  
 global yrot  
 global zrot  
 global cube\_z\_coor  
 global ambient  
 global lightpos  
  
 scale\_tetra\_Y= 1.25  
 scale\_tetra\_Z = 1.25  
 scale\_tetra\_X = 1.25  
 scale\_tetra\_Mod = 0.75  
 pot\_pos = -10  
 sphere\_pos = 0  
 rot = 5  
  
 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 = (0.4, 0.4, 0.4, 1) # Первые три числа цвет в формате RGB, а последнее - яркость  
 lightpos = (0.0, 1.0, 1.0) # Положение источника освещения по осям xyz  
  
 glClearColor(0.5, 0.5, 0.5, 0.0) # Серый цвет для первоначальной закраски  
  
 # glColor3f(0.0, 0.0, 0.0); черный  
 # glColor3f(1.0, 0.0, 0.0); красный  
 # glColor3f(0.0, 1.0, 0.0); зеленый  
 # glColor3f(1.0, 1.0, 0.0); желтый  
 # glColor3f(0.0, 0.0, 1.0); синий  
 # glColor3f(1.0, 0.0, 1.0); фиолетовый  
 # glColor3f(0.0, 1.0, 1.0); голубой  
 # glColor3f(1.0, 1.0, 1.0); белый  
  
 gluOrtho2D(-200.0, 200.0, -200.0, 200.0) # 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  
  
  
# Процедура обработки специальных клавиш  
  
  
def specialkeys(key, x, y):  
 global scale\_tetra\_Y  
 global scale\_tetra\_Z  
 global scale\_tetra\_X  
 global scale\_tetra\_Mod  
 global pot\_pos  
 global sphere\_pos  
 global rot  
 global xrot  
 global yrot  
 global zrot  
 global cube\_z\_coor  
 global lin\_att  
 global quad\_att  
 global lightpos  
 global lin\_att  
 global quad\_att  
 # Обработчики для клавиш со стрелками  
 if key == GLUT\_KEY\_UP: # Клавиша вверх  
 # xrot -= 2.0 # Уменьшаем угол вращения по оси Х  
 scale\_tetra\_X \*= scale\_tetra\_Mod  
 scale\_tetra\_Y \*= scale\_tetra\_Mod  
 scale\_tetra\_Z \*= scale\_tetra\_Mod  
 glutPostRedisplay()  
 if key == GLUT\_KEY\_DOWN: # Клавиша вниз  
 # xrot += 2.0 # Увеличиваем угол вращения по оси Х  
 scale\_tetra\_X /= scale\_tetra\_Mod  
 scale\_tetra\_Y /= scale\_tetra\_Mod  
 scale\_tetra\_Z /= scale\_tetra\_Mod  
 glutPostRedisplay()  
 if key == GLUT\_KEY\_LEFT: # Клавиша влево  
 # yrot -= 2.0 # Уменьшаем угол вращения по оси Y  
 pot\_pos += 1  
 sphere\_pos -= 1  
 rot += 1  
 glutPostRedisplay()  
 if key == GLUT\_KEY\_RIGHT: # Клавиша вправо  
 # yrot += 2.0 # Увеличиваем угол вращения по оси Y  
 pot\_pos -= 1  
 sphere\_pos += 1  
 rot -= 1  
 glutPostRedisplay()  
 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  
  
 if key == GLUT\_KEY\_F10:  
 zrot += 2.0  
 if key == GLUT\_KEY\_F9:  
 zrot -= 2.0  
 if key == GLUT\_KEY\_F8:  
 xrot += 2.0  
 if key == GLUT\_KEY\_F7:  
 xrot -= 2.0  
 if key == GLUT\_KEY\_F6:  
 yrot += 2.0  
 if key == GLUT\_KEY\_F5:  
 yrot -= 2.0  
 # if key == GLUT\_KEY\_F7:  
 # # zrot+=10.0  
 # if key == GLUT\_KEY\_F8:  
 # # cube\_z\_coor+=0.01  
 # if key == GLUT\_KEY\_F9:  
 # # cube\_z\_coor-=0.01  
  
  
  
  
 glutPostRedisplay() # Вызываем процедуру перерисовки  
  
  
# Процедура перерисовки  
def draw():  
 global xrot  
 global yrot  
 global lightpos  
 global cube\_z  
  
 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 с фигурами  
  
 # Устанавливаем материал: рисовать с 2 сторон, рассеянное освещение, коричневый цвет  
 # glRotatef(30, 0.0, 0.0, 1.0)  
 # glTranslatef(0.0, 0.0, -cube\_z\_coor)  
 # glTranslatef(0.0, 0.0, 0.0)  
  
 glClearColor(0.1, 0.1, 0.1, 0.0)  
 # glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT)  
 glMatrixMode(GL\_MODELVIEW)  
 glLoadIdentity()  
 glTranslatef(2.0, 0.0, -100.0)  
  
 glPushMatrix()  
 glColor3f(0.0, 1.0, 0.0)  
 glRotatef(0, 0, 0, 0)  
 glTranslated(pot\_pos, 25, 0)  
 glutWireTeapot(10)  
 glPopMatrix()  
  
   
  
 glPushMatrix()  
 glColor3f(1.0, 0.0, 0.0)  
 glRotatef(90, 0, 0, 0)  
 glTranslated(0, -20, -20)  
  
 cylinder = gluNewQuadric()  
 gluQuadricDrawStyle(cylinder, GLU\_SILHOUETTE)  
 gluCylinder(cylinder, 15, 15, 50, 50, 100)  
  
  
  
 glPopMatrix() # Возвращаем сохраненное положение "камеры"  
 glutSwapBuffers() # Выводим все нарисованное в памяти на экран  
  
def reshape (width, height):  
 global aspect  
 # int height  
 # int width  
  
 if height == 0: height = 1  
 aspect = width / height  
 glViewport(0, 0, width, height)  
 glMatrixMode(GL\_PROJECTION)  
 glLoadIdentity();  
 gluPerspective(60.0, aspect, 0.1, 200.0) #glOrtho  
  
  
  
# 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(900, 200)  
  
# initialize opengl  
glutInit(sys.argv)  
  
# create the main window  
glutCreateWindow("Lab1\_Baganov")  
  
# redraw  
glutDisplayFunc(draw)  
  
# this procedure handles keystrokes  
glutSpecialFunc(specialkeys)  
# glutDisplayFunc(renderScene);  
glutReshapeFunc(reshape);  
# glutIdleFunc(renderScene);  
# glutSpecialFunc(control\_proc);  
# call the initialization function  
init()  
  
# run the main loop  
glutMainLoop()