



Министерство науки и высшего образования Российской Федерации  
Калужский филиал  
федерального государственного бюджетного  
образовательного учреждения высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(КФ МГТУ им. Н.Э. Баумана)

**ФАКУЛЬТЕТ ИУК «Информатика и управление»**

**КАФЕДРА ИУК4 «Программное обеспечение ЭВМ, информационные технологии»**

## **ЛАБОРАТОРНАЯ РАБОТА № 2**

### **«Основные примитивы OpenGL»**

**ДИСЦИПЛИНА: «Компьютерная графика»**

Выполнил: студент гр. ИУК4-41Б

\_\_\_\_\_ Дубовицкий Д.А.  
(Подпись)

Проверил:

\_\_\_\_\_ Глебов С. А.  
(Подпись)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Калуга, 2023

**Цель:** Целью выполнения лабораторной работы является формирование практических навыков по работе с графическими примитивами OpenGL, а также применения к ним эффектов средствами машины состояний и использования проверки глубины.

**Задание:**

1. научиться устанавливать размеры наблюдаемого объема
2. изучить параметры функции `glVertex`
3. изучить основные параметры функции `glBegin`
4. сформировать понимание особенности использования функции `glEnable` с конкретными геометрическими примитивами
5. выяснить основы построения сплошных объектов

**Задание 1**

Для [Листинга 1](#) задать с использованием тригонометрических преобразований (функций `sin` и `cos`) произвольное множество точек (не менее 30 точек) в трехмерном пространстве. Изучить принципы вращения экрана с помощью клавиш курсора и способы подключения меню к программе.

**Листинг программы:**

```
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import math
import numpy as np
# window size
width = 700
height = 700

xRot = 0
yRot = 0

def ChangeSize(w, h):
    nRange = 100
    if h == 0:
        h = 1
    glViewport(0, 0, w, h)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    if w <= h:
        glOrtho(-nRange, nRange, -nRange * h / w, nRange * h / w, -nRange, nRange)
    else:
        glOrtho(-nRange * w / h, nRange * w / h, -nRange, nRange, -nRange, nRange)

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

def SpecialKeys(key, x, y):
    global xRot, yRot
    if key == GLUT_KEY_UP:
```

```

        xRot -= 5
    if key == GLUT_KEY_DOWN:
        xRot += 5
    if key == GLUT_KEY_LEFT:
        yRot -= 5
    if key == GLUT_KEY_RIGHT:
        yRot += 5
    glutPostRedisplay()

def display():
    angle = 0
    glClear(GL_COLOR_BUFFER_BIT)
    glPushMatrix()
    glRotatef(xRot, 1, 0, 0)
    glRotatef(yRot, 0, 1, 0)
    glPointSize(10)
    glBegin(GL_POINTS)
    z = -100
    while angle <= 100:
        angle += 0.1
        x = 50 * math.cos(angle)
        y = 50 * math.sin(angle)
        glVertex3f(x, y, z)
        z += 0.5
    glEnd()
    glPopMatrix()
    glutSwapBuffers()

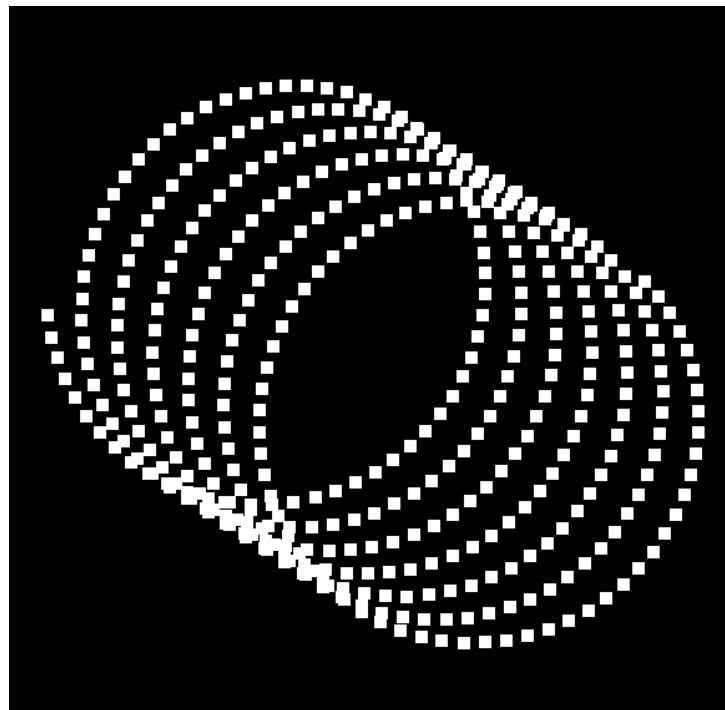
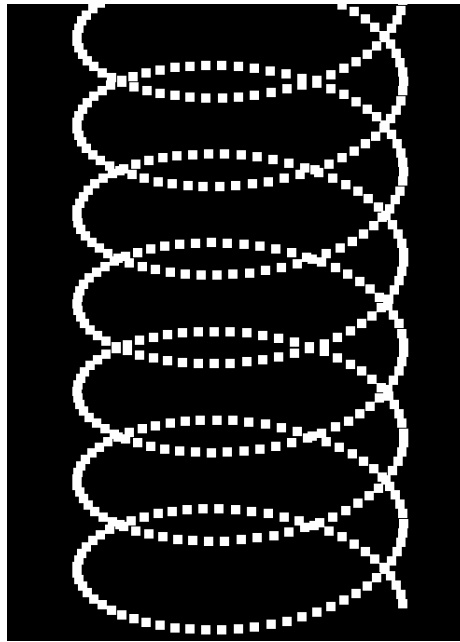
# initialize window
glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH)
glutInitWindowSize(width, height)
glutCreateWindow("Graphics")
glutReshapeFunc(ChangeSize)
glutSpecialFunc(SpecialKeys)
# set background color
glClearColor(0, 0, 0, 0.0)

# set callback functions
glutDisplayFunc(display)

# start main loop
glutMainLoop()

```

**Результаты выполнения:**



## Задание 2

Для [Листинга 2](#) задать произвольное множество точек разного размера и цвета в трехмерном пространстве.

### Листинг программы

```
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import math
import numpy as np
import random
```

```

# window size
width = 700
height = 700

xRot = 0
yRot = 0

def ChangeSize(w, h):
    nRange = 100
    if h == 0:
        h = 1
    glViewport(0, 0, w, h)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    if w <= h:
        glOrtho(-nRange, nRange, -nRange * h / w, nRange * h / w, -nRange, nRange)
    else:
        glOrtho(-nRange * w / h, nRange * w / h, -nRange, nRange, -nRange, nRange)

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

def SpecialKeys(key, x, y):
    global xRot, yRot
    if key == GLUT_KEY_UP:
        xRot -= 5
    if key == GLUT_KEY_DOWN:
        xRot += 5
    if key == GLUT_KEY_LEFT:
        yRot -= 5
    if key == GLUT_KEY_RIGHT:
        yRot += 5
    glutPostRedisplay()

def display():
    angle = 0
    siz = 1
    glClear(GL_COLOR_BUFFER_BIT)
    glPushMatrix()
    glRotatef(xRot, 1, 0, 0)
    glRotatef(yRot, 0, 1, 0)
    z = -100
    while angle <= 100:
        angle += 0.1
        glPointSize(siz)
        x = 50 * math.cos(angle)
        y = 50 * math.sin(angle)
        glBegin(GL_POINTS)
        glColor3f(random.uniform(0.0, 1.0), random.uniform(0.0, 1.0), random.uniform(0.0,
1.0))
        glVertex3f(x, y, z)
        glEnd()
        siz += 0.1
        z += 0.5
    glPopMatrix()
    glutSwapBuffers()

# initialize window
glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH)

```

```

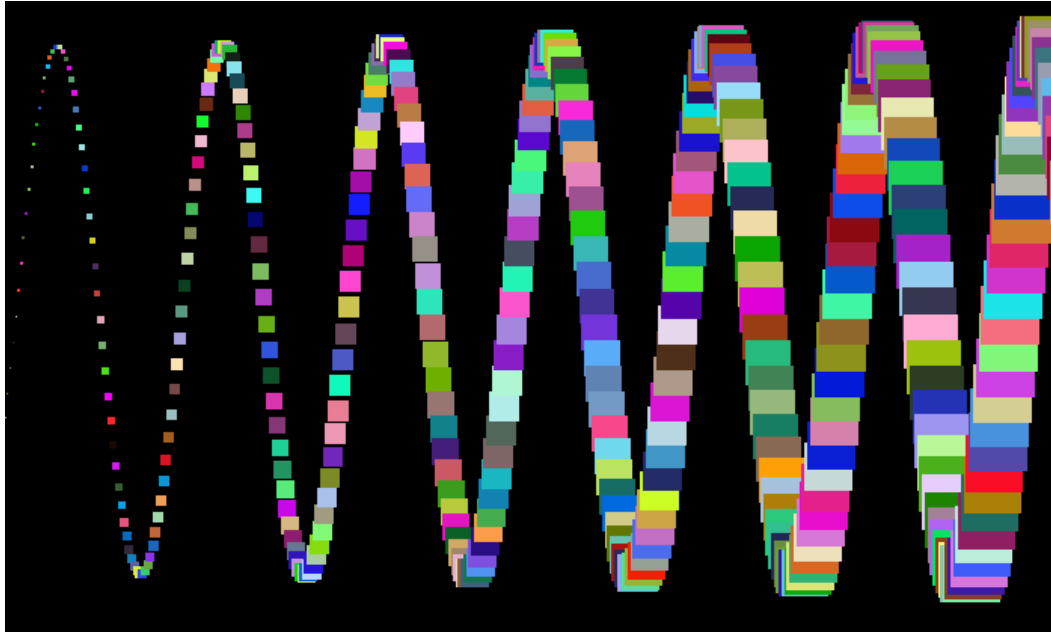
glutInitWindowSize(width, height)
glutCreateWindow("Graphics")
glutReshapeFunc(ChangeSize)
glutSpecialFunc(SpecialKeys)
# set background color
glClearColor(0, 0, 0, 0.0)

# set callback functions
glutDisplayFunc(display)

# start main loop
glutMainLoop()

```

### Результаты выполнения:



### Задание 3

Для [Листинга 3](#) изобразить не менее 10 линий с разными параметрами (цвет, толщина, начертание по шаблону) в трехмерном пространстве.

#### Листинг программы:

```

from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import random
xRot = 0.0
yRot = 0.0

def RenderScene():
    glEnable(GL_LINE_STIPPLE)
    patterns = [0x0F0F, 0xFFFF, 0xF0F0, 0xCCCC, 0xCF CF, 0xCCFF, 0xAAAA, 0xCAFC, 0xACAB, 0xFCA0]
    wid = 1
    startY = -90.0
    glClear(GL_COLOR_BUFFER_BIT)
    glGetFloatv(GL_LINE_WIDTH_RANGE, wid)
    for i in range(10):

```

```

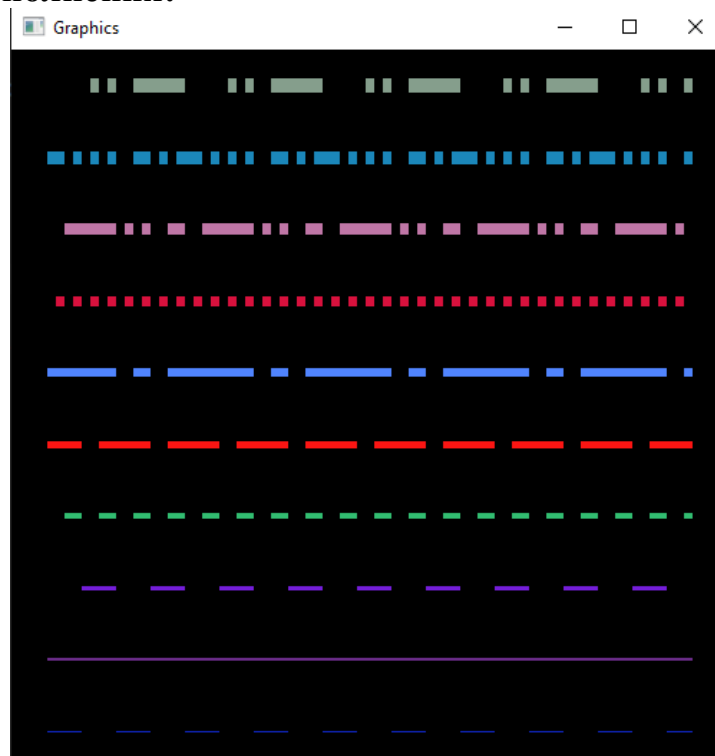
glLineStipple(6, patterns[i])
glLineWidth(wid)
glColor3f(random.uniform(0.0, 1.0), random.uniform(0.0, 1.0), random.uniform(0.0, 1.0))
glBegin(GL_LINES)
glVertex2f(-90.0, startY + i * 20)
glVertex2f(90.0, startY + i * 20)
glEnd()
wid += 1
glutSwapBuffers()

def ChangeSize(w,h):
    nRange = 100.0
    if h == 0:
        h = 1
    glViewport(0, 0, w, h)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    if w <= h:
        glOrtho(-nRange, nRange, -nRange * h / w, nRange * h / w, -nRange, nRange)
    else:
        glOrtho(-nRange * w / h, nRange * w / h, -nRange, nRange, -nRange, nRange)
    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH)
glutInitWindowSize(500, 500)
glutCreateWindow("Graphics")
glutReshapeFunc(ChangeSize)
# set background color
glClearColor(0, 0, 0, 0.0)
# set callback functions
glutDisplayFunc(RenderScene)
# start main loop
glutMainLoop()

```

## Результаты выполнения:



## Задание 4

Для [Листинга 4](#) выполнить аппроксимацию произвольной кривой линии посредством коротких прямых. На 3 примерах наглядно продемонстрировать точность аппроксимации. По возможности использовать меню.

### Листинг программы

```
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import math

xRot = 0.0
yRot = 0.0

def RenderScene():
    angle = 0
    glClear(GL_COLOR_BUFFER_BIT)
    glPushMatrix()
    glRotatef(xRot, 1.0, 0.0, 0.0)
    glRotatef(yRot, 0.0, 1.0, 0.0)
    glBegin(GL_LINE_STRIP)
    z = -50.0
    while angle <= 50:
        angle += 0.3
        x = 50 * math.cos(angle)
        y = 50 * math.sin(angle)
        glVertex3f(x, y, z)
        z += 0.5
    glEnd()
    glPopMatrix()
    glutSwapBuffers()

def SpecialKeys(key, x, y):
    global xRot, yRot
    if key == GLUT_KEY_UP:
        xRot -= 5
    if key == GLUT_KEY_DOWN:
        xRot += 5
    if key == GLUT_KEY_LEFT:
        yRot -= 5
    if key == GLUT_KEY_RIGHT:
        yRot += 5
    glutPostRedisplay()

def ChangeSize(w, h):
    nRange = 100
    if h == 0:
        h = 1
    glViewport(0, 0, w, h)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    if w <= h:
        glOrtho(-nRange, nRange, -nRange * h / w, nRange * h / w, -nRange, nRange)
    else:
        glOrtho(-nRange * w / h, nRange * w / h, -nRange, nRange, -nRange, nRange)
```



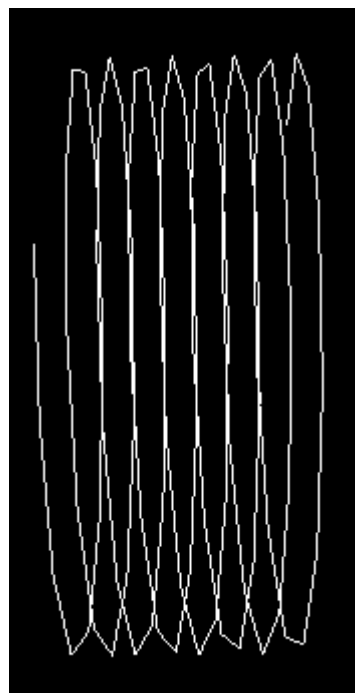
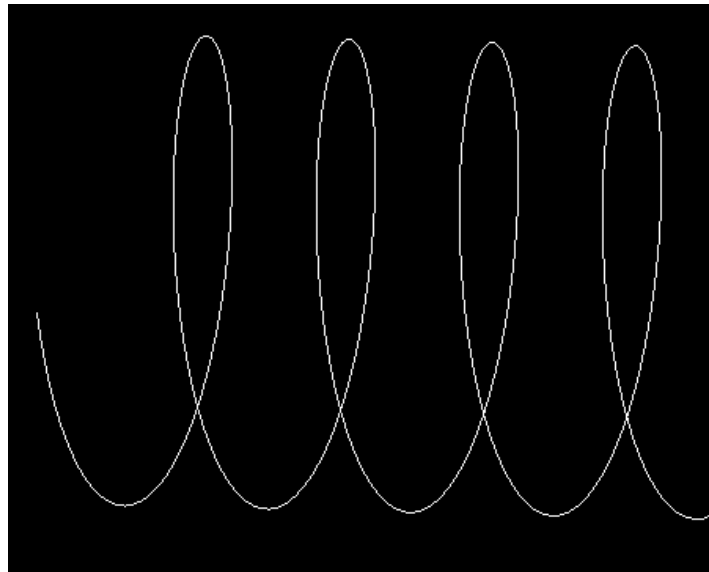
```

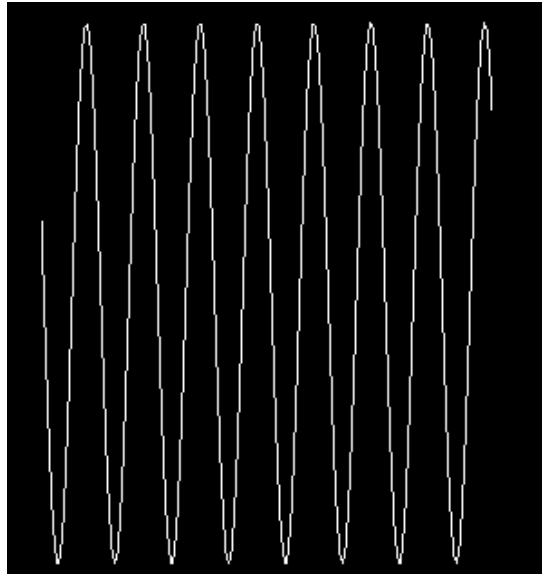
glMatrixMode(GL_MODELVIEW)
glLoadIdentity()

glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH)
glutInitWindowSize(600, 600)
glutCreateWindow("Graphics")
glutReshapeFunc(ChangeSize)
glutSpecialFunc(SpecialKeys)
glClearColor(0, 0, 0, 0.0)
glutDisplayFunc(RenderScene)
glutMainLoop()

```

## Результаты выполнения





## Задание 5

Для [Листинга 5](#), используя `GL_LINE_STRIP` и `GL_LINE_LOOP` нарисовать трехмерный объект из ломанных линий содержащих не менее 15 точек. Точки задавать, используя цикл.

### Листинг программы

```
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import math
xRot = 0.0
yRot = 0.0

def RenderScene():
    glClear(GL_COLOR_BUFFER_BIT)
    glPushMatrix()
    glRotatef(xRot, 1.0, 0.0, 0.0)
    glRotatef(yRot, 0.0, 1.0, 0.0)
    angle = 0
    # Круг
    glBegin(GL_LINE_STRIP)
    while angle <= 15:
        z = 0.5
        angle += 0.1
        x = 50 * math.cos(angle)
        y = 50 * math.sin(angle)
        glColor3f(1.0, 0.0, 0.0)
        glVertex3f(x, y, z)
    glEnd()
    # Конус
    glBegin(GL_LINE_LOOP)
    angle = 0
    while angle <= 15:
        z = 0.5
        angle += 0.5
        x = 50 * math.cos(angle)
        y = 50 * math.sin(angle)
```

```

        glColor3f(0.0, 0.0, 1.0)
        glVertex3f(x, y, z)
        glVertex3f(0, 0, -50)
    glEnd()

    glPopMatrix()
    glutSwapBuffers()

def SpecialKeys(key, x, y):
    global xRot, yRot
    if key == GLUT_KEY_UP:
        xRot -= 5
    if key == GLUT_KEY_DOWN:
        xRot += 5
    if key == GLUT_KEY_LEFT:
        yRot -= 5
    if key == GLUT_KEY_RIGHT:
        yRot += 5
    glutPostRedisplay()

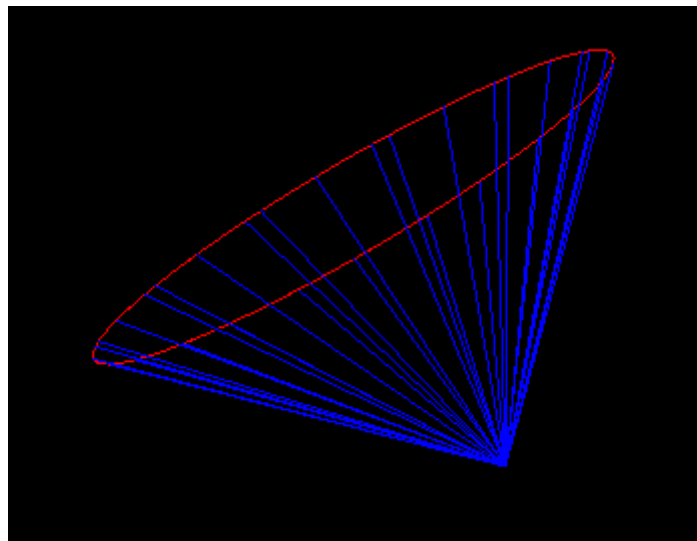
def ChangeSize(w, h):
    nRange = 100
    if h == 0:
        h = 1
    glViewport(0, 0, w, h)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    if w <= h:
        glOrtho(-nRange, nRange, -nRange * h / w, nRange * h / w, -nRange, nRange)
    else:
        glOrtho(-nRange * w / h, nRange * w / h, -nRange, nRange, -nRange, nRange)

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH)
glutInitWindowSize(600, 600)
glutCreateWindow("Graphics")
glutReshapeFunc(ChangeSize)
glutSpecialFunc(SpecialKeys)
glClearColor(0, 0, 0, 0.0)
glutDisplayFunc(RenderScene)
glutMainLoop()

```

## Результаты выполнения



## Задание 6

Для [Листинга 6](#) используя GL\_TRIANGLES вывести на экран произвольную трехмерную геометрическую фигуру состоящую из треугольников. Понимать принципы обхода точек треугольника и полигонов и знать на что он влияет.

### Листинг программы:

```
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import math

xRot = 0.0
yRot = 0.0

def RenderScene():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glEnable(GL_DEPTH_TEST)
    glRotatef(xRot, 1.0, 0.0, 0.0)
    glRotatef(yRot, 0.0, 1.0, 0.0)
    glBegin(GL_TRIANGLES)

    glColor3f(0.0, 1.0, 0.0)
    glVertex3f(0, 0, 0)
    glColor3f(0.0, 1.0, 0.0)
    glVertex3f(30, 0, 0)
    glColor3f(0.0, 1.0, 0.0)
    glVertex3f(15, 30, -20)

    glColor3f(1.0, 0.0, 0.0)
    glVertex3f(30, 0, 0)
    glColor3f(1.0, 0.0, 0.0)
    glVertex3f(30, 0, -30)
    glColor3f(1.0, 0.0, 0.0)
    glVertex3f(15, 30, -20)

    glColor3f(0.0, 0.0, 1.0)
    glVertex3f(0, 0, -30)
    glColor3f(0.0, 0.0, 1.0)
    glVertex3f(30, 0, -30)
    glColor3f(0.0, 0.0, 1.0)
    glVertex3f(15, 30, -20)

    glColor3f(1.0, 0.5, 0.0)
    glVertex3f(0, 0, 0)
    glColor3f(1.0, 0.5, 0.0)
    glVertex3f(0, 0, -30)
    glColor3f(1.0, 0.5, 0.0)
    glVertex3f(15, 30, -20)
    glEnd()

    glBegin(GL_QUADS)
    glColor3f(1.0, 0.0, 1.0)
    glVertex3f(0, 0, 0)
    glColor3f(1.0, 0.0, 1.0)
    glVertex3f(30, 0, 0)
    glColor3f(1.0, 0.0, 1.0)
    glVertex3f(30, 0, -30)
```

```

glColor3f(1.0, 0.0, 1.0)
glVertex3f(0, 0, -30)

glEnd()

glutSwapBuffers()

def SpecialKeys(key, x, y):
    global xRot, yRot
    if key == GLUT_KEY_UP:
        xRot -= 5
    if key == GLUT_KEY_DOWN:
        xRot += 5
    if key == GLUT_KEY_LEFT:
        yRot -= 5
    if key == GLUT_KEY_RIGHT:
        yRot += 5
    glutPostRedisplay()

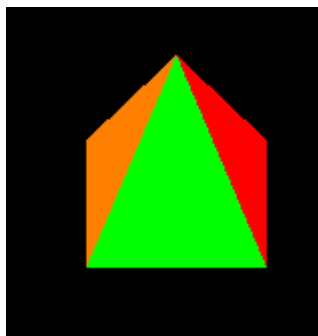
def ChangeSize(w, h):
    nRange = 100
    if h == 0:
        h = 1
    glViewport(0, 0, w, h)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    if w <= h:
        glOrtho(-nRange, nRange, -nRange * h / w, nRange * h / w, -nRange, nRange)
    else:
        glOrtho(-nRange * w / h, nRange * w / h, -nRange, nRange, -nRange, nRange)

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH)
glutInitWindowSize(600, 600)
glutCreateWindow("Graphics")
glutReshapeFunc(ChangeSize)
glutSpecialFunc(SpecialKeys)
glClearColor(0, 0, 0, 0.0)
glutDisplayFunc(RenderScene)
glutMainLoop()

```

### Результаты выполнения:



### Задание 7

Для [Листинга 7](#) продемонстрировать работу директив [GL\\_TRIANGLE\\_STRIP](#) и [GL\\_TRIANGLE\\_FAN](#).

### Листинг программы:

```
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import math

xRot = 0.0
yRot = 0.0

def RenderScene():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glEnable(GL_DEPTH_TEST)
    glRotatef(xRot, 1.0, 0.0, 0.0)
    glRotatef(yRot, 0.0, 1.0, 0.0)
    glBegin(GL_TRIANGLE_FAN)
    z = -50.0
    glVertex3f(0, 0, 0)
    currentColor = False
    for i in range(100):
        if currentColor:
            glColor3f(0.0, 0.0, 0.5)
        else:
            glColor3f(0.0, 0.0, 1.0)
        currentColor = not currentColor
        x = 50.0 * math.sin(i)
        y = 50.0 * math.cos(i)
        glVertex3f(x, y, z)
    glEnd()
    glBegin(GL_TRIANGLE_FAN)
    glVertex3f(0, 0, -100)
    for i in range(100):
        if currentColor:
            glColor3f(0.0, 0.0, 1.0)
        else:
            glColor3f(0.0, 0.0, 0.5)
        currentColor = not currentColor
        x = 50.0 * math.sin(i)
        y = 50.0 * math.cos(i)
        glVertex3f(x, y, z)
    glEnd()
    glutSwapBuffers()

def SpecialKeys(key, x, y):
    global xRot, yRot
    if key == GLUT_KEY_UP:
        xRot -= 5
    if key == GLUT_KEY_DOWN:
        xRot += 5
    if key == GLUT_KEY_LEFT:
        yRot -= 5
    if key == GLUT_KEY_RIGHT:
        yRot += 5
    glutPostRedisplay()

def ChangeSize(w, h):
    nRange = 100
    if h == 0:
        h = 1
    glViewport(0, 0, w, h)
    glMatrixMode(GL_PROJECTION)
```

```

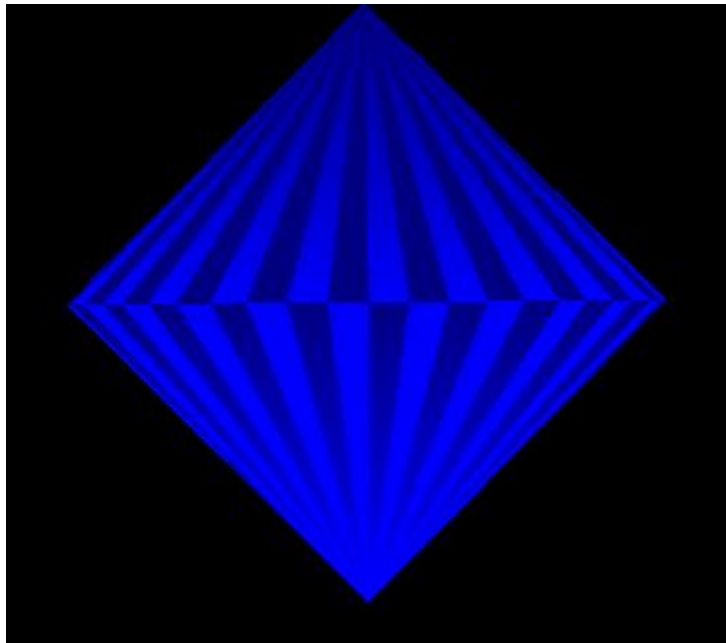
glLoadIdentity()
if w <= h:
    glOrtho(-nRange, nRange, -nRange * h / w, nRange * h / w, -nRange, nRange)
else:
    glOrtho(-nRange * w / h, nRange * w / h, -nRange, nRange, -nRange, nRange)

glMatrixMode(GL_MODELVIEW)
glLoadIdentity()

glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH)
glutInitWindowSize(600, 600)
glutCreateWindow("Graphics")
glutReshapeFunc(ChangeSize)
glutSpecialFunc(SpecialKeys)
glClearColor(0, 0, 0, 0.0)
glutDisplayFunc(RenderScene)
glutMainLoop()

```

### Результаты выполнения:



### Листинг программы:

```

from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import math

xRot = 0.0
yRot = 0.0

def RenderScene():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glEnable(GL_DEPTH_TEST)
    glRotatef(xRot, 1.0, 0.0, 0.0)
    glRotatef(yRot, 0.0, 1.0, 0.0)
    glBegin(GL_TRIANGLE_STRIP)
    z = -50.0
    currentColor = False
    for i in range(50):

```

```

        if currentColor:
            glColor3f(0.0, 0.0, 0.5)
        else:
            glColor3f(0.0, 0.0, 1.0)
        currentColor = not currentColor
        x = 50.0 * math.sin(i)
        y = 50.0 * math.cos(i)
        glVertex3f(x, y, z)
    glEnd()
    glutSwapBuffers()

def SpecialKeys(key, x, y):
    global xRot, yRot
    if key == GLUT_KEY_UP:
        xRot -= 5
    if key == GLUT_KEY_DOWN:
        xRot += 5
    if key == GLUT_KEY_LEFT:
        yRot -= 5
    if key == GLUT_KEY_RIGHT:
        yRot += 5
    glutPostRedisplay()

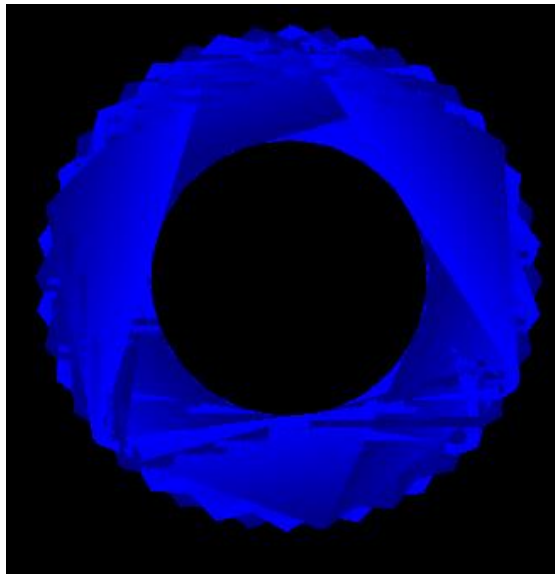
def ChangeSize(w, h):
    nRange = 100
    if h == 0:
        h = 1
    glViewport(0, 0, w, h)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    if w <= h:
        glOrtho(-nRange, nRange, -nRange * h / w, nRange * h / w, -nRange, nRange)
    else:
        glOrtho(-nRange * w / h, nRange * w / h, -nRange, nRange, -nRange, nRange)

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH)
glutInitWindowSize(600, 600)
glutCreateWindow("Graphics")
glutReshapeFunc(ChangeSize)
glutSpecialFunc(SpecialKeys)
glClearColor(0, 0, 0, 0.0)
glutDisplayFunc(RenderScene)
glutMainLoop()

```





## Задание 8

Для [Листинга 8](#) для произвольно заданной фигуры составленной из не менее чем 6 треугольников продемонстрировать работу функции `glShadeModel`.

## Листинг программы

```
from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import math

xRot = 0.0
yRot = 0.0

fly = [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0xc0,
        0x00, 0x00, 0x01, 0xf0, 0x00, 0x00, 0x07, 0xf0, 0x0f, 0x00, 0x1f, 0xe0, 0x1f, 0x80,
        0x1f, 0xc0, 0x0f, 0xc0, 0x3f, 0x80,
        0x07, 0xe0, 0x7e, 0x00, 0x03, 0xf0, 0xff, 0x80, 0x03, 0xf5, 0xff, 0xe0, 0x07, 0xfd,
        0xff, 0xf8, 0x1f, 0xfc, 0xff, 0xe8,
        0xff, 0xe3, 0xbf, 0x70, 0xde, 0x80, 0xb7, 0x00, 0x71, 0x10, 0x4a, 0x80, 0x03, 0x10,
        0x4e, 0x40, 0x02, 0x88, 0x8c, 0x20,
        0x05, 0x05, 0x04, 0x40, 0x02, 0x82, 0x14, 0x40, 0x02, 0x40, 0x10, 0x80, 0x02, 0x64,
        0x1a, 0x80, 0x00, 0x92, 0x29, 0x00,
        0x00, 0xb0, 0x48, 0x00, 0x00, 0xc8, 0x90, 0x00, 0x00, 0x85, 0x10, 0x00, 0x00, 0x03,
        0x00, 0x00, 0x00, 0x00, 0x10, 0x00]

def RenderScene():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glEnable(GL_DEPTH_TEST)
    glPolygonMode(GL_BACK, GL_LINE)
    glRotatef(xRot, 1.0, 0.0, 0.0)
    glRotatef(yRot, 0.0, 1.0, 0.0)
    glBegin(GL_TRIANGLE_FAN)
    z = -50.0
    glVertex3f(0, 0, 0)
    currentColor = False
    for i in range(100):
```

```

        if currentColor:
            glColor3f(0.0, 0.0, 0.5)
        else:
            glColor3f(0.0, 0.0, 1.0)
        currentColor = not currentColor
        x = 50.0 * math.sin(i)
        y = 50.0 * math.cos(i)
        glVertex3f(x, y, z)
    glEnd()
    glBegin(GL_TRIANGLE_FAN)
    glVertex3f(0, 0, -100)
    for i in range(100):
        if currentColor:
            glColor3f(0.0, 0.0, 1.0)
        else:
            glColor3f(0.0, 0.0, 0.5)
        currentColor = not currentColor
        x = 50.0 * math.sin(i)
        y = 50.0 * math.cos(i)
        glVertex3f(x, y, z)
    glEnd()

    glutSwapBuffers()

def SpecialKeys(key, x, y):
    global xRot, yRot
    if key == GLUT_KEY_UP:
        xRot -= 5
    if key == GLUT_KEY_DOWN:
        xRot += 5
    if key == GLUT_KEY_LEFT:
        yRot -= 5
    if key == GLUT_KEY_RIGHT:
        yRot += 5
    glutPostRedisplay()

def SetupRC():
    glShadeModel(GL_FLAT)
    glEnable(GL_POLYGON_STIPPLE)
    glPolygonStipple(fly)

def ChangeSize(w, h):
    nRange = 100
    if h == 0:
        h = 1
    glViewport(0, 0, w, h)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    if w <= h:
        glOrtho(-nRange, nRange, -nRange * h / w, nRange * h / w, -nRange, nRange)
    else:
        glOrtho(-nRange * w / h, nRange * w / h, -nRange, nRange, -nRange, nRange)

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH)
glutInitWindowSize(600, 600)
glutCreateWindow("Graphics")

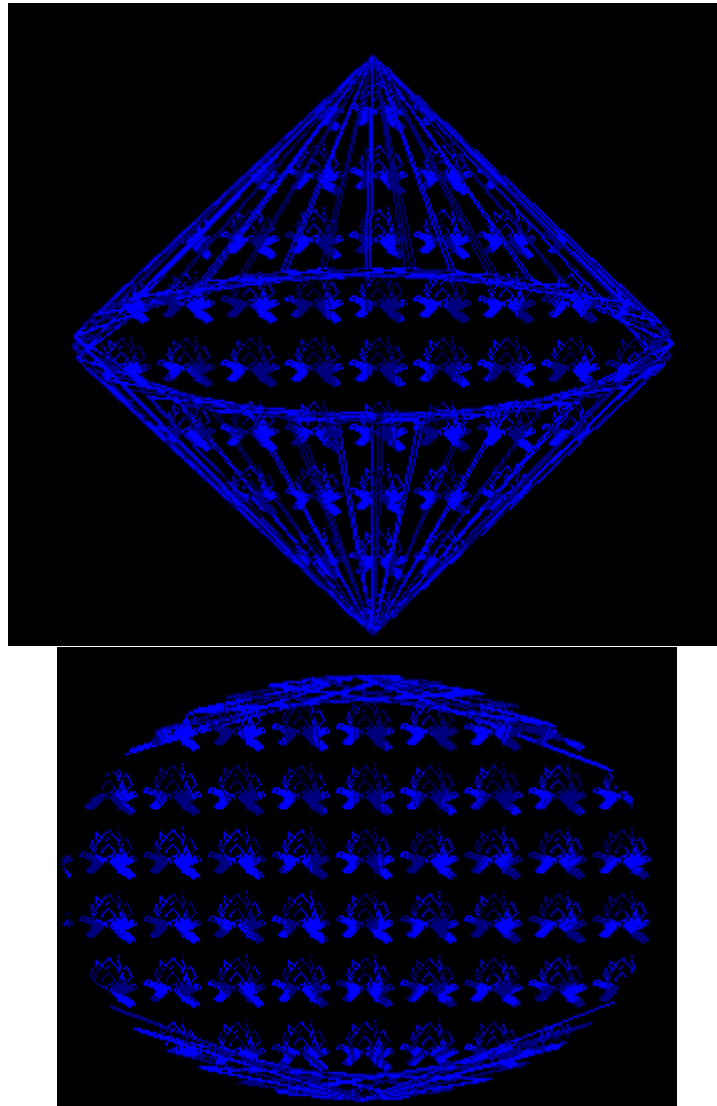
```

```

glutReshapeFunc(ChangeSize)
glutSpecialFunc(SpecialKeys)
glClearColor(0, 0, 0, 0.0)
glutDisplayFunc(RenderScene)
SetupRC()
glutMainLoop()

```

**Результаты выполнения:**



## Задание 9

Для [Листинга 9](#) на наглядном примере продемонстрировать работу проверки глубины для технологии удаления скрытых поверхностей. На наглядном примере продемонстрировать работу технологии отбора задних граней.

### Листинг программы:

```

from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import math

```

```

xRot = 0.0
yRot = 0.0

```

```

def RenderScene():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glEnable(GL_DEPTH_TEST)
    glEnable(GL_CULL_FACE)
    glCullFace(GL_BACK)

    glRotatef(xRot, 1.0, 0.0, 0.0)
    glRotatef(yRot, 0.0, 1.0, 0.0)

    glBegin(GL_TRIANGLES)

    glColor3f(0.0, 1.0, 0.0)
    glVertex3f(0, 0, 0)
    glColor3f(0.0, 1.0, 0.0)
    glVertex3f(30, 0, 0)
    glColor3f(0.0, 1.0, 0.0)
    glVertex3f(15, 30, -20)

    glColor3f(1.0, 0.0, 0.0)
    glVertex3f(30, 0, 0)
    glColor3f(1.0, 0.0, 0.0)
    glVertex3f(30, 0, -30)
    glColor3f(1.0, 0.0, 0.0)
    glVertex3f(15, 30, -20)

    glColor3f(0.0, 0.0, 1.0)
    glVertex3f(0, 0, -30)
    glColor3f(0.0, 0.0, 1.0)
    glVertex3f(30, 0, -30)
    glColor3f(0.0, 0.0, 1.0)
    glVertex3f(15, 30, -20)

    glColor3f(1.0, 0.5, 0.0)
    glVertex3f(0, 0, 0)
    glColor3f(1.0, 0.5, 0.0)
    glVertex3f(0, 0, -30)
    glColor3f(1.0, 0.5, 0.0)
    glVertex3f(15, 30, -20)
    glEnd()

    glBegin(GL_QUADS)
    glColor3f(1.0, 0.0, 1.0)
    glVertex3f(0, 0, 0)
    glColor3f(1.0, 0.0, 1.0)
    glVertex3f(30, 0, 0)
    glColor3f(1.0, 0.0, 1.0)
    glVertex3f(30, 0, -30)
    glColor3f(1.0, 0.0, 1.0)
    glVertex3f(0, 0, -30)

    glEnd()

    glutSwapBuffers()

def SpecialKeys(key, x, y):
    global xRot, yRot
    if key == GLUT_KEY_UP:
        xRot -= 5
    if key == GLUT_KEY_DOWN:
        xRot += 5

```

```

    if key == GLUT_KEY_LEFT:
        yRot -= 5
    if key == GLUT_KEY_RIGHT:
        yRot += 5
    glutPostRedisplay()

def SetupRC():
    glShadeModel(GL_FLAT)

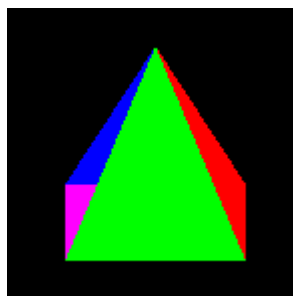
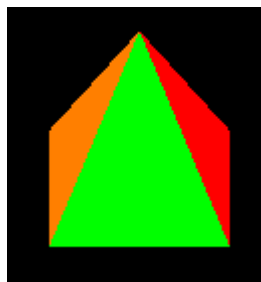
def ChangeSize(w, h):
    nRange = 100
    if h == 0:
        h = 1
    glViewport(0, 0, w, h)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    if w <= h:
        glOrtho(-nRange, nRange, -nRange * h / w, nRange * h / w, -nRange, nRange)
    else:
        glOrtho(-nRange * w / h, nRange * w / h, -nRange, nRange, -nRange, nRange)

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH)
glutInitWindowSize(600, 600)
glutCreateWindow("Graphics")
glutReshapeFunc(ChangeSize)
glutSpecialFunc(SpecialKeys)
glClearColor(0, 0, 0, 0.0)
glutDisplayFunc(RenderScene)
SetupRC()
glutMainLoop()

```

**Результаты выполнения:**



**Листинг программы:**

```

from OpenGL.GL import *
from OpenGL.GLUT import *
from OpenGL.GLU import *
import math
xRot = 0.0
yRot = 0.0

def RenderScene():
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT)
    glEnable(GL_DEPTH_TEST)
    glPolygonMode(GL_BACK, GL_LINE)
    glRotatef(xRot, 1.0, 0.0, 0.0)
    glRotatef(yRot, 0.0, 1.0, 0.0)
    glBegin(GL_TRIANGLES)

    glColor3f(0.0, 1.0, 0.0)
    glVertex3f(0, 0, 0)
    glColor3f(0.0, 1.0, 0.0)
    glVertex3f(30, 0, 0)
    glColor3f(0.0, 1.0, 0.0)
    glVertex3f(15, 30, -20)

    glColor3f(1.0, 0.0, 0.0)
    glVertex3f(30, 0, 0)
    glColor3f(1.0, 0.0, 0.0)
    glVertex3f(30, 0, -30)
    glColor3f(1.0, 0.0, 0.0)
    glVertex3f(15, 30, -20)

    glColor3f(0.0, 0.0, 1.0)
    glVertex3f(0, 0, -30)
    glColor3f(0.0, 0.0, 1.0)
    glVertex3f(30, 0, -30)
    glColor3f(0.0, 0.0, 1.0)
    glVertex3f(15, 30, -20)

    glColor3f(1.0, 0.5, 0.0)
    glVertex3f(0, 0, 0)
    glColor3f(1.0, 0.5, 0.0)
    glVertex3f(0, 0, -30)
    glColor3f(1.0, 0.5, 0.0)
    glVertex3f(15, 30, -20)
    glEnd()

    glBegin(GL_QUADS)
    glColor3f(1.0, 0.0, 1.0)
    glVertex3f(0, 0, 0)
    glColor3f(1.0, 0.0, 1.0)
    glVertex3f(30, 0, 0)
    glColor3f(1.0, 0.0, 1.0)
    glVertex3f(30, 0, -30)
    glColor3f(1.0, 0.0, 1.0)
    glVertex3f(0, 0, -30)

    glEnd()

    glutSwapBuffers()

def SpecialKeys(key, x, y):
    global xRot, yRot
    if key == GLUT_KEY_UP:

```

```

    xRot -= 5
if key == GLUT_KEY_DOWN:
    xRot += 5
if key == GLUT_KEY_LEFT:
    yRot -= 5
if key == GLUT_KEY_RIGHT:
    yRot += 5
glutPostRedisplay()

def SetupRC():
    glShadeModel(GL_FLAT)

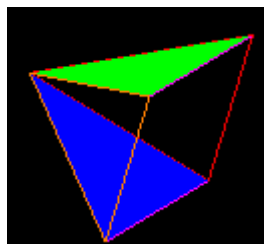
def ChangeSize(w, h):
    nRange = 100
    if h == 0:
        h = 1
    glViewport(0, 0, w, h)
    glMatrixMode(GL_PROJECTION)
    glLoadIdentity()
    if w <= h:
        glOrtho(-nRange, nRange, -nRange * h / w, nRange * h / w, -nRange, nRange)
    else:
        glOrtho(-nRange * w / h, nRange * w / h, -nRange, nRange, -nRange, nRange)

    glMatrixMode(GL_MODELVIEW)
    glLoadIdentity()

glutInit()
glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH)
glutInitWindowSize(600, 600)
glutCreateWindow("Graphics")
glutReshapeFunc(ChangeSize)
glutSpecialFunc(SpecialKeys)
glClearColor(0, 0, 0, 0.0)
glutDisplayFunc(RenderScene)
SetupRC()
glutMainLoop()

```

### Результаты выполнения:



**Вывод:** в ходе выполнения лабораторной работы были сформированы практические навыки по работе с графическими примитивами OpenGL, а также применения к ним эффектов средствами машины состояний и использования проверки глубины. Изучены: параметры функции `glVertex`, основные параметры функции `glBegin`, особенности использования функции `glEnable` с конкретными геометрическими примитивами, основы построения сплошных объектов.