

**КАЗАНСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. А. Н. ТУПОЛЕВА – КАИ**

Институт компьютерных технологий и защиты информации

Кафедра: Автоматизированных систем обработки информации и управления

Г.М. НАБЕРЕЖНОВ, Н.Н. МАКСИМОВ

ЭЛЕМЕНТАРНОЕ ПРОГРАММИРОВАНИЕ ГРАФИКИ В OpenGL

**Методическое пособие
к лабораторным работам по курсу
«Компьютерная геометрия и графика»**

Студент, гр. 4210

Гауиш М.Г

Преподаватель

Гаптуллазянова Гульшат Ильдусовна

Казань – 2023

Лабораторная работа N°3 (Отчет)

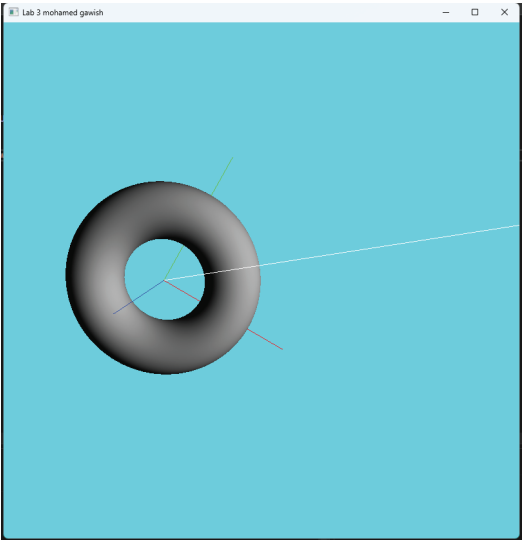
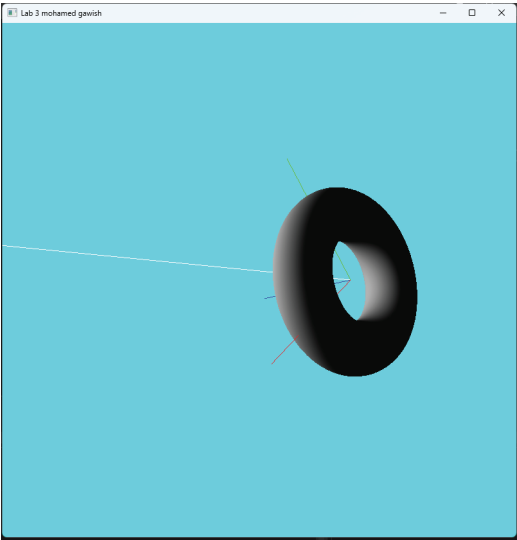
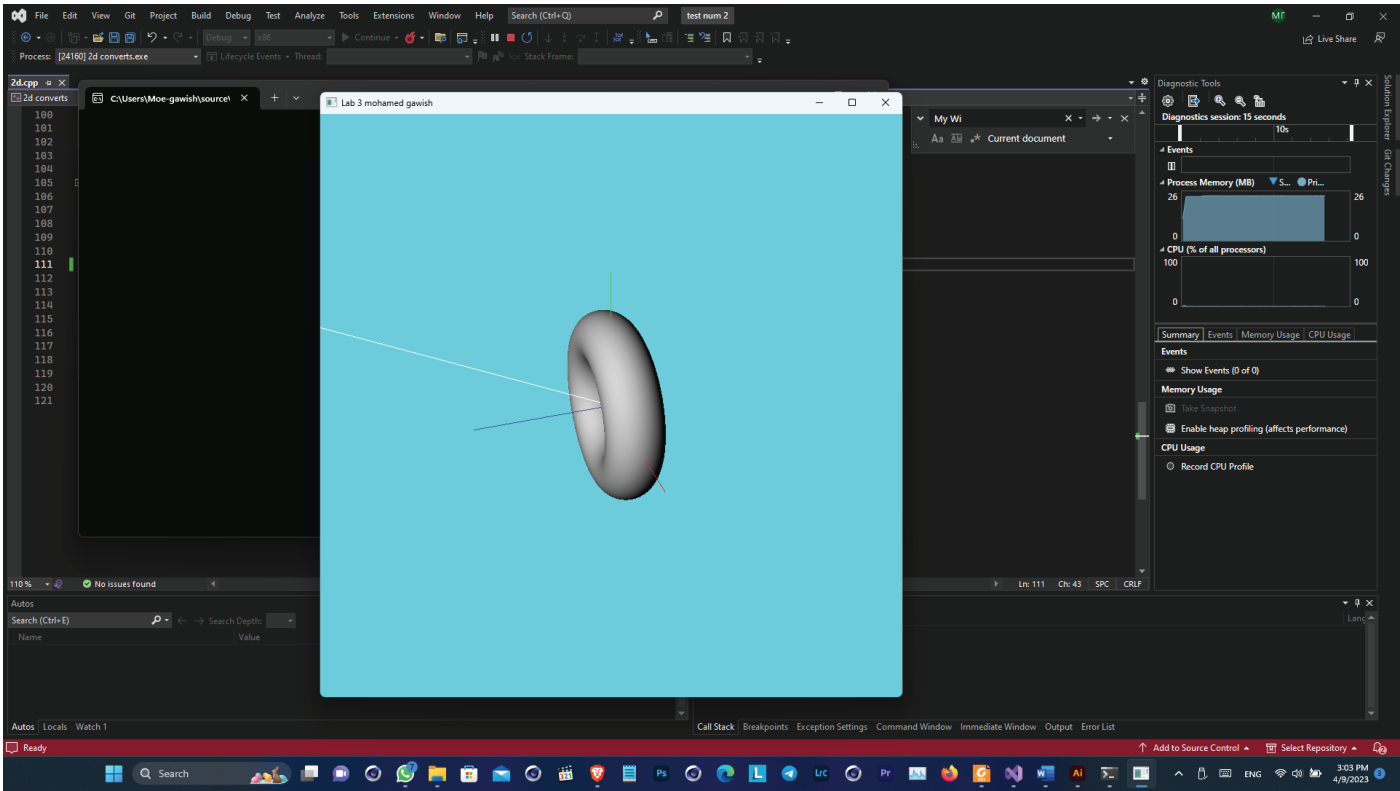
ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИ

Создайте программу, где геометрический примитив из библиотеки GLUT помещен в начало МСК, освещен неподвижным источником и в программе реализованы функции обработки нажатий клавиатуры и клавиш мыши в соответствии с таблицей.

Вариант 2

2	тор	кл. X - наблюдатель вращается в пл. (осьY, E) вокруг объекта по ч.с. по кл. Y - наблюдатель вращается в пл. (осьY, E) против ч.с.	левая кл. — объект масштабируется с увеличением размеров левая кл. — объект масштабируется с уменьшением размеров
---	-----	--	--

Скриншот работы программы:



```

1  #include <GL/glut.h>
2  #include <cmath>
3
4  GLfloat angle = 0;
5  GLfloat scale = 1.0; // added scale factor
6  void init(void)
7  {
8      glClearColor(0.0, 1.0, 1.0, 0.0);
9      glMatrixMode(GL_PROJECTION);
10     glLoadIdentity();
11     gluPerspective(60, 1, 1, 10);
12     glMatrixMode(GL_MODELVIEW);
13     glLoadIdentity();
14 }
15 void myDisplay()
16 {
17     glPushMatrix(); //Сохраняем VM = 1
18     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
19     glEnable(GL_DEPTH_TEST);
20     gluLookAt(3, 2, 1, 0, 0, 0, 0, 1, 0); // VM=Fwe
21     GLfloat myLightPosition[] = { 3.0, 3.0, 3.0, 1.0 }; // Источник света в СКw
22     glLightfv(GL_LIGHT0, GL_POSITION, myLightPosition); /*Позиция источника света будет преобразована в СКe*/
23     glEnable(GL_LIGHTING);
24     glEnable(GL_LIGHT0);
25     glPushMatrix(); //Сохраняем VM=Fwe
26     glRotatef(angle, 0, 1, 0); // VM=Fwe*R
27     glRotatef(angle, 0, 1, 0); // VM=Fwe*R*R
28     glRotatef(angle, 0, 0, 1); // VM=Fwe*R*R*R
29     glScalef(scale, scale, scale); // added scaling
30     glutSolidTorus(0.2, 0.5, 100, 100);
31     glPopMatrix(); // Восстанавливаем VM=Fwe
32     glDisable(GL_LIGHTING); //Выключаем освещение
33     glBegin(GL_LINES);
34     glColor3f(1, 0, 0); glVertex3f(0, 0, 0); glVertex3f(1, 0, 0);
35     glColor3f(0, 1, 0); glVertex3f(0, 0, 0); glVertex3f(0, 1, 0);
36     glColor3f(0, 0, 1); glVertex3f(0, 0, 0); glVertex3f(0, 0, 1);
37     glEnd();
38     glBegin(GL_LINES);
39     glColor3f(1, 1, 1); glVertex3f(3, 3, 3); glVertex3f(0.0, 0.0, 0.0);
40     glEnd();
41     glPopMatrix();
42
43     glutSwapBuffers();
44 }
45 void myReshape(int width, int height)
46 {
47     if (width / height < 1) glViewport(0, 0, width, width);
48     else glViewport(0, 0, height, height);
49 }
50
51 void myIdle()
52 {
53     angle += 0.5;
54     if (angle > 360.0) angle = 0;
55     glutPostRedisplay();
56     Sleep(1);
57 }
58
59 void myKeyboard(unsigned char key, int x, int y)
60 {
61     switch (key)
62     {
63     case 'x':
64         // Наблюдатель вращается в пл. (осьY, E) вокруг объекта по ч.с.
65         glMatrixMode(GL_MODELVIEW);
66         glTranslatef(0, 0, -3);
67         glRotatef(2, 0, 1, 0);
68         glTranslatef(0, 0, 3);
69         glutPostRedisplay();
70         break;
71     case 'y':
72         // Наблюдатель вращается в пл. (осьY, E) против ч.с.
73         glMatrixMode(GL_MODELVIEW);
74         glTranslatef(0, 0, -3);
75         glRotatef(-2, 0, 1, 0);
76         glTranslatef(0, 0, 3);
77         glutPostRedisplay();
78         break;
79     default:
80         break;
81     }
82 }
83
84 void myMouse(int button, int state, int x, int y)
85 {
86     switch (button)
87     {
88

```

```

89         case GLUT_LEFT_BUTTON:
90             // Объект масштабируется с увеличением размеров
91             glScalef(1.1, 1.1, 1.1);
92             glutPostRedisplay();
93             break;
94         case GLUT_RIGHT_BUTTON:
95             // Объект масштабируется с уменьшением размеров
96             glScalef(0.9, 0.9, 0.9);
97             glutPostRedisplay();
98             break;
99
100        default:
101            break;
102    }
103 }
104
105 int main(int argc, char* argv[])
106 {
107     glutInit(&argc, argv);
108     glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
109     glutInitWindowSize(800, 800);
110     glutInitWindowPosition(0, 0);
111     glutCreateWindow("Lab 3 mohamed gawish");
112     glutDisplayFunc(myDisplay);
113     glutReshapeFunc(myReshape);
114     // glutIdleFunc(myIdle);
115     init();
116     glutKeyboardFunc(myKeyboard);
117     glutMouseFunc(myMouse);
118     glutMainLoop();
119 }

```

```

#include <GL/glut.h>
#include <cmath>

GLfloat angle = 0;
GLfloat scale = 1.0; // added scale factor
void init(void)
{
    glClearColor(0.0, 1.0, 1.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(10, 1, 1, 60);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
void myDisplay()
{
    glPushMatrix(); //Сохраняем VM = 1
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_DEPTH_TEST);
    gluLookAt(0, 1, 0, 0, 0, 0, 1, 2, 3); // VM=Fwe
    GLfloat myLightPosition[] = { 1.0, 3.0, 3.0, 3.0 }; // Источник света в СКw
    glLightfv(GL_LIGHT0, GL_POSITION, myLightPosition); /*Позиция источника света будет преобразована в СКe*/
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glPushMatrix(); //Сохраняем VM=Fwe
    glRotatef(angle, 0, 1, 0); // VM=Fwe*R
    glRotatef(angle, 0, 1, 0); // VM=Fwe*R*R
    glRotatef(angle, 1, 0, 0); // VM=Fwe*R*R*R
    glScalef(scale, scale, scale); // added scaling
    glutSolidTorus(100, 100, 0.5, 0.2);
    glPopMatrix(); // Восстанавливаем VM=Fwe
    glDisable(GL_LIGHTING); //Выключаем освещение
    glBegin(GL_LINES);
    glColor3f(0, 0, 1); glVertex3f(0, 0, 0); glVertex3f(0, 0, 1);
    glColor3f(0, 1, 0); glVertex3f(0, 0, 0); glVertex3f(0, 1, 0);
    glColor3f(1, 0, 0); glVertex3f(0, 0, 0); glVertex3f(1, 0, 0);
    glEnd();
    glBegin(GL_LINES);
    glColor3f(1, 1, 1); glVertex3f(3, 3, 3); glVertex3f(0.0, 0.0, 0.0);
    glEnd();
    glPopMatrix();

    glutSwapBuffers();
}
void myReshape(int width, int height)
{
    if (width / height < 1) glViewport(0, 0, width, width);
    else glViewport(0, 0, height, height);
}
void myIdle()
{
    angle += 0.5;
    if (angle > 360.0) angle = 0;
    glutPostRedisplay();
    Sleep(1);
}
void myKeyboard(unsigned char key, int x, int y)
{
    switch (key)
    {
        case 'x':

```

```

// Наблюдатель вращается в пл. (осьY, E) вокруг объекта по ч.с.
glMatrixMode(GL_MODELVIEW);
glTranslatef(3- ,0 ,0);
glRotatef(0 ,1 ,0 ,2);
glTranslatef(3 ,0 ,0);
glutPostRedisplay();
break;
case 'y':
// Наблюдатель вращается в пл. (осьY, E) против ч.с.
glMatrixMode(GL_MODELVIEW);
glTranslatef(3- ,0 ,0);
glRotatef(-0 ,1 ,0 ,2);
glTranslatef(3 ,0 ,0);
glutPostRedisplay();
break;

default:
break;
}
}

```

```

void myMouse(int button, int state, int x, int y)
{
switch (button)
{
case GLUT_LEFT_BUTTON:
// Объект масштабиру-ется с увеличением размеров
glScalef(1.1 ,1.1 ,1.1);
glutPostRedisplay();
break;
case GLUT_RIGHT_BUTTON:
// Объект масштабиру-ется с уменьшением размеров
glScalef(0.9 ,0.9 ,0.9);
glutPostRedisplay();
break;

default:
break;
}
}

```

```

int main(int argc, char* argv[])
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE | GLUT_DEPTH);
glutInitWindowSize(800 ,800);
glutInitWindowPosition(0 ,0);
glutCreateWindow("Lab 3 mohamed gawish");
glutDisplayFunc(myDisplay);
glutReshapeFunc(myReshape);
// glutIdleFunc(myIdle);
init();
glutKeyboardFunc(myKeyboard);
glutMouseFunc(myMouse);
glutMainLoop();
}

```

Библиотека Glut и opengl использовалась.

Большое спасибо)