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 №3 (Отчет)

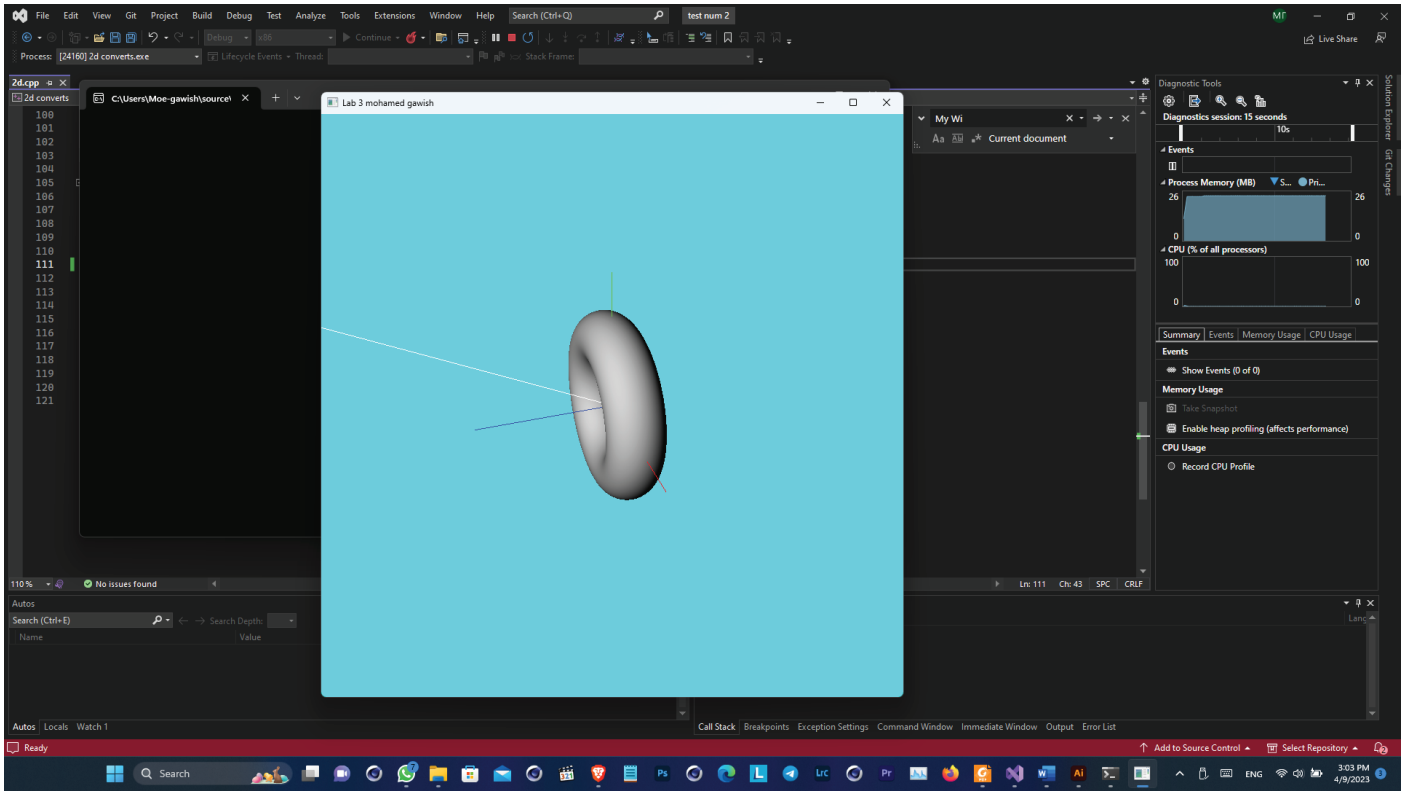
ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИ

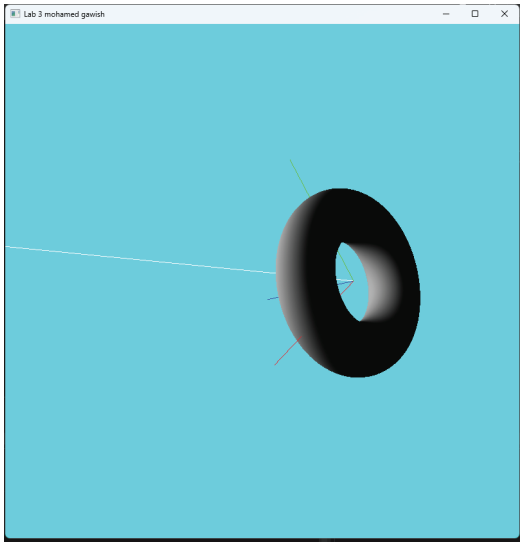
Создайте программу, где геометрический примитив из библиотеки GLUT помещен в начало МСК, освещен неподвижным источником и в программе реализованы функции обработки нажатий клавиатуры и клавиш мыши в соответствии с таблицей.

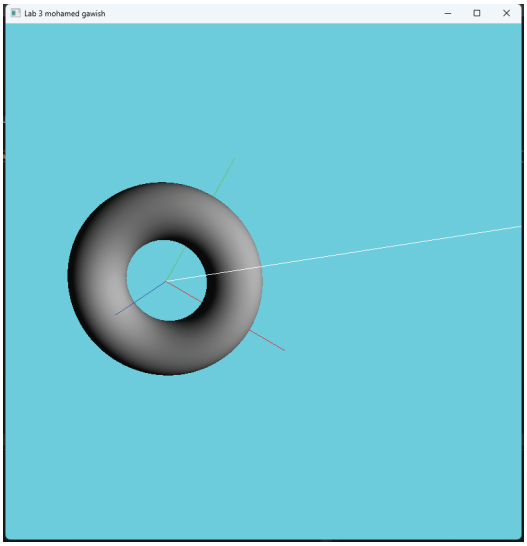
Вариант 2

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:







```c
#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(60, 1, 1, 10);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
void myDisplay()
{
    glPushMatrix(); //Сохраняем VM = 1
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glEnable(GL_DEPTH_TEST);
    gluLookAt(3, 2, 1, 0, 0, 0, 0, 1, 0); // VM=Fwe
    GLfloat myLightPosition[] = { 3.0, 3.0, 3.0, 1.0 }; // Источник света в СКw
    glLightfv(GL_LIGHT0, GL_POSITION, myLightPosition); /*Позиция источника света будет преобразована в СКе*/
    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, 0, 0, 1); // VM=Fwe*R*R*R
    glScalef(scale, scale, scale); // added scaling
    glutSolidTorus(0.2, 0.5, 100, 100);
    glPopMatrix(); // Восстанавливаем VM=Fwe
    glDisable(GL_LIGHTING); //Выключаем освещение
    glBegin(GL_LINES);
    glColor3f(1, 0, 0); glVertex3f(0, 0, 0); glVertex3f(1, 0, 0);
    glColor3f(0, 1, 0); glVertex3f(0, 0, 0); glVertex3f(0, 1, 0);
    glColor3f(0, 0, 1); glVertex3f(0, 0, 0); glVertex3f(0, 0, 1);
    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(0, 0, -3);
        glRotatef(2, 0, 1, 0);
        glTranslatef(0, 0, 3);
        glutPostRedisplay();
        break;
    case 'y':
        // Наблюдатель вращается в пл. (осьY, E) против ч.с.
        glMatrixMode(GL_MODELVIEW);
        glTranslatef(0, 0, -3);
        glRotatef(-2, 0, 1, 0);
        glTranslatef(0, 0, 3);
        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();
}
```

```
#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)