**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное бюджетное образовательное учреждение высшего**

**профессионального образования**

**«Казанский национальный исследовательский технический университет**

**им. А.Н. Туполева-КАИ»**

Институт компьютерных технологий и защиты информации

Лабораторная работа № 4

по дисциплине «Компьютерная графика»

**Выполнил**

студент группы 4211

Абдуллаев Л.Х.

Казань

2020

# Задание 1

Цель :

1) Создать приложение на основе библиотеки (GLUT) OpenGL Utility Toolkit, которое открывает окно для рисования геометрического объекта. При этом режим дисплея использует двойную буферизацию, режим RGBA и буфер глубины.

2) Установить перспективную проекцию с углом обзора 60.

3) Расположить наблюдателя в позицию (0, 3, 3), направленного в точку (0, 0, 0) и вектором направления вверх (0, 1, 0).

4) Выполнить рисование тетраэдра, используя пример из листинга 1.

5) Создать локальный источник света, используя пример из листинга 2.

6) Осуществить вращение геометрического объекта (источника света) вокруг оси Y, при помощи обработчика отсутствия событий. При нажатии на клавишу ‘o’(‘l’) – вращается объект (источник света).

7) Осуществить визуализацию лицевых (нелицевых) граней. При нажатии на клавишу ‘f’(‘b’) – визуализируются лицевые (нелицевые) грани.

Листинг:

#include <math.h>

#include <Gl\glut.h>

GLfloat angle = 0;

GLfloat plusangle = 0;

GLfloat angle2 = 0;

GLfloat plusangle2 = 0;

int flagO = 1;

int flagL = 1;

void init(void)

{

glClearColor(1.0, 1.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(60, 1, 1, 10);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void myDisplay()

{

glPushMatrix();

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glEnable(GL\_DEPTH\_TEST);

gluLookAt(3, 3, 3, 0, 0, 0, 0, 1, 0);

GLfloat myLightPosition[] = { 1.0, 2.0, 2.0, 1.0 };

glLightfv(GL\_LIGHT0, GL\_POSITION, myLightPosition);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glPushMatrix();

glRotatef(angle2, 0, 1, 0);

glLightfv(GL\_LIGHT0, GL\_POSITION, myLightPosition);

glPopMatrix();

glPushMatrix();

glRotatef(angle, 0, 1, 0);

// грань f0

glBegin(GL\_POLYGON);

glNormal3f(0.577, 0.577, 0.577);

glVertex3f(1, 0, 0);

glVertex3f(0, 1, 0);

glVertex3f(0, 0, 1);

glEnd();

// грань f1

glBegin(GL\_POLYGON);

glNormal3f(0, 0, -1);

glVertex3f(0, 0, 0);

glVertex3f(0, 1, 0);

glVertex3f(1, 0, 0);

glEnd();

// грань f2

glBegin(GL\_POLYGON);

glNormal3f(-1, 0, 0);

glVertex3f(0, 0, 0);

glVertex3f(0, 0, 1);

glVertex3f(0, 1, 0);

glEnd();

// грань f3

glBegin(GL\_POLYGON);

glNormal3f(0, -1, 0);

glVertex3f(1, 0, 0);

glVertex3f(0, 0, 1);

glVertex3f(0, 0, 0);

glEnd();

glPopMatrix();

glDisable(GL\_LIGHTING);

glBegin(GL\_LINES);

glColor3f(1, 0, 0); glVertex3f(0, 0, 0); glVertex3f(2, 0, 0);

glColor3f(0, 1, 0); glVertex3f(0, 0, 0); glVertex3f(0, 2, 0);

glColor3f(0, 0, 1); glVertex3f(0, 0, 0); glVertex3f(0, 0, 2);

glEnd();

glRotatef(angle2, 0, 1, 0);

glBegin(GL\_LINES);

glColor3f(1, 1, 1);

glVertex3f(1, 2, 2);

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 += plusangle;

if (angle > 360.0)

angle = 0;

angle2 += plusangle2;

if (angle2 > 360.0)

angle2 = 0;

glutPostRedisplay();

}

void keys(unsigned char key, int x, int y)

{

if (key == 'O' || key == 'o')

{

if (flagO == 1)

plusangle = 0.2;

else

plusangle = 0;

flagO \*= -1;

}

else if (key == 'L' || key == 'l')

{

if (flagL == 1)

plusangle2 = 0.2;

else

plusangle2 = 0;

flagL \*= -1;

}

if (key == 'F' || key == 'f')

{

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

}

else if (key == 'B' || key == 'b')

{

glCullFace(GL\_FRONT);

glEnable(GL\_CULL\_FACE);

}

}

int main(int argc, char\* argv[])

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGBA | GLUT\_DOUBLE | GLUT\_DEPTH);

glutInitWindowPosition(0, 0);

glutCreateWindow("tetrahedron");

glutDisplayFunc(myDisplay);

glutKeyboardFunc(keys);

glutReshapeFunc(myReshape);

glutIdleFunc(myIdle);

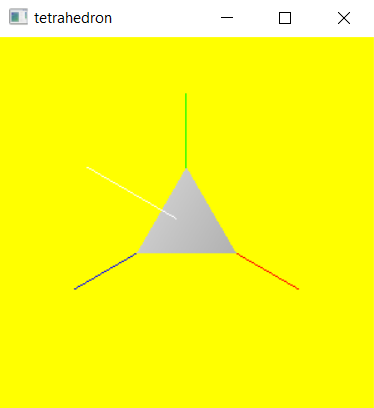
init();

glutMainLoop();

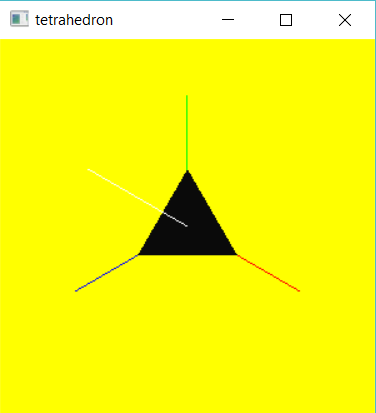
}

Результат работы:

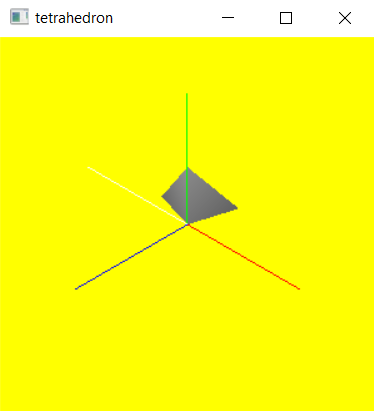
Начальное положение.



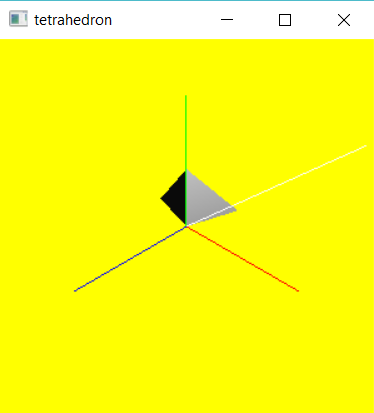
Визуализация нелицевых граней.



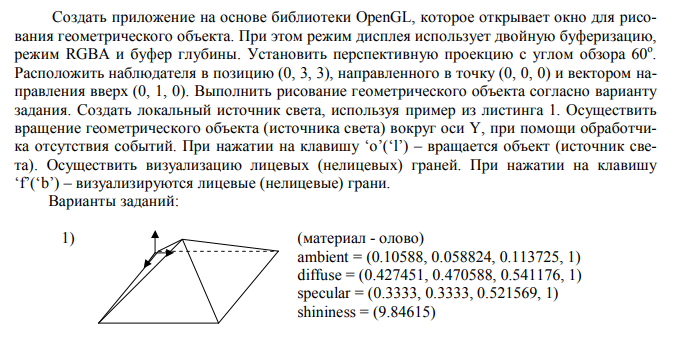
Фигура повернута.



Повернут источник света.



# Задание 2. Вариант 1.



Листинг:

#include <math.h>

#include <Gl\glut.h>

GLfloat angle = 0;

GLfloat plusangle = 0;

GLfloat angle2 = 0;

GLfloat plusangle2 = 0;

int flagO = 1;

int flagL = 1;

void init(void)

{

glClearColor(1.0, 1.0, 0.0, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(60, 1, 1, 10);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

}

void myDisplay()

{

glPushMatrix();

glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

glEnable(GL\_DEPTH\_TEST);

gluLookAt(3, 3, 3, 0, 0, 0, 0, 1, 0);

GLfloat myLightPosition[] = { 1.0, 2.0, 2.0, 1.0 };

GLfloat myDiffuse[] = { 0.427451, 0.470588, 0.541176, 1 };

glMaterialfv(GL\_FRONT, GL\_DIFFUSE, myDiffuse);

GLfloat myShininess[] = { 9.84615 };

glMaterialfv(GL\_FRONT, GL\_SHININESS, myShininess);

GLfloat myAmbient[] = { 0.10588, 0.058824, 0.113725, 1 };

glMaterialfv(GL\_FRONT, GL\_AMBIENT, myAmbient);

GLfloat mySpecular[] = { 0.3333, 0.3333, 0.521569, 1 };

glMaterialfv(GL\_FRONT, GL\_SPECULAR, mySpecular);

glLightfv(GL\_LIGHT0, GL\_POSITION, myLightPosition);

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glPushMatrix();

glRotatef(angle2, 0, 1, 0);

glBegin(GL\_LINES);

glColor3f(1, 1, 1);

glVertex3f(1, 2, 2);

glVertex3f(0.0, 0.0, 0.0);

glEnd();

glLightfv(GL\_LIGHT0, GL\_POSITION, myLightPosition);

glPopMatrix();

glPushMatrix();

glRotatef(angle, 0, 1, 0);

glBegin(GL\_POLYGON);

glNormal3f(0, 0.557, 0.557);

glVertex3f(0, 0, 1.5);

glVertex3f(1.5, 0, 1.5);

glVertex3f(0.75, 1, 0.75);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(-0.557, 0.557, 0);

glVertex3f(0, 0, 0);

glVertex3f(0, 0, 1.5);

glVertex3f(0.75, 1, 0.75);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, 0.557, -0.557);

glVertex3f(1.5, 0, 0);

glVertex3f(0, 0, 0);

glVertex3f(0.75, 1, 0.75);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0.557, 0.557, 0);

glVertex3f(1.5, 0, 1.5);

glVertex3f(1.5, 0, 0);

glVertex3f(0.75, 1, 0.75);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, -1, 0);

glVertex3f(0, 0, 0);

glVertex3f(1.5, 0, 0);

glVertex3f(1.5, 0, 1.5);

glVertex3f(0, 0, 1.5);

glEnd();

glPopMatrix();

glDisable(GL\_LIGHTING);

glBegin(GL\_LINES);

glColor3f(1, 0, 0); glVertex3f(0, 0, 0); glVertex3f(2, 0, 0);

glColor3f(0, 1, 0); glVertex3f(0, 0, 0); glVertex3f(0, 2, 0);

glColor3f(0, 0, 1); glVertex3f(0, 0, 0); glVertex3f(0, 0, 2);

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 += plusangle;

if (angle > 360.0)

angle = 0;

angle2 += plusangle2;

if (angle2 > 360.0)

angle2 = 0;

glutPostRedisplay();

}

void keys(unsigned char key, int x, int y)

{

if (key == 'O' || key == 'o')

{

if (flagO == 1)

plusangle = 0.2;

else

plusangle = 0;

flagO \*= -1;

}

else if (key == 'L' || key == 'l')

{

if (flagL == 1)

plusangle2 = 0.2;

else

plusangle2 = 0;

flagL \*= -1;

}

if (key == 'F' || key == 'f')

{

glCullFace(GL\_BACK);

glEnable(GL\_CULL\_FACE);

}

else if (key == 'B' || key == 'b')

{

glCullFace(GL\_FRONT);

glEnable(GL\_CULL\_FACE);

}

}

int main(int argc, char\* argv[])

{

glutInit(&argc, argv);

glutInitDisplayMode(GLUT\_RGBA | GLUT\_DOUBLE | GLUT\_DEPTH);

glutInitWindowPosition(0, 0);

glutCreateWindow("tetrahedron");

glutDisplayFunc(myDisplay);

glutKeyboardFunc(keys);

glutReshapeFunc(myReshape);

glutIdleFunc(myIdle);

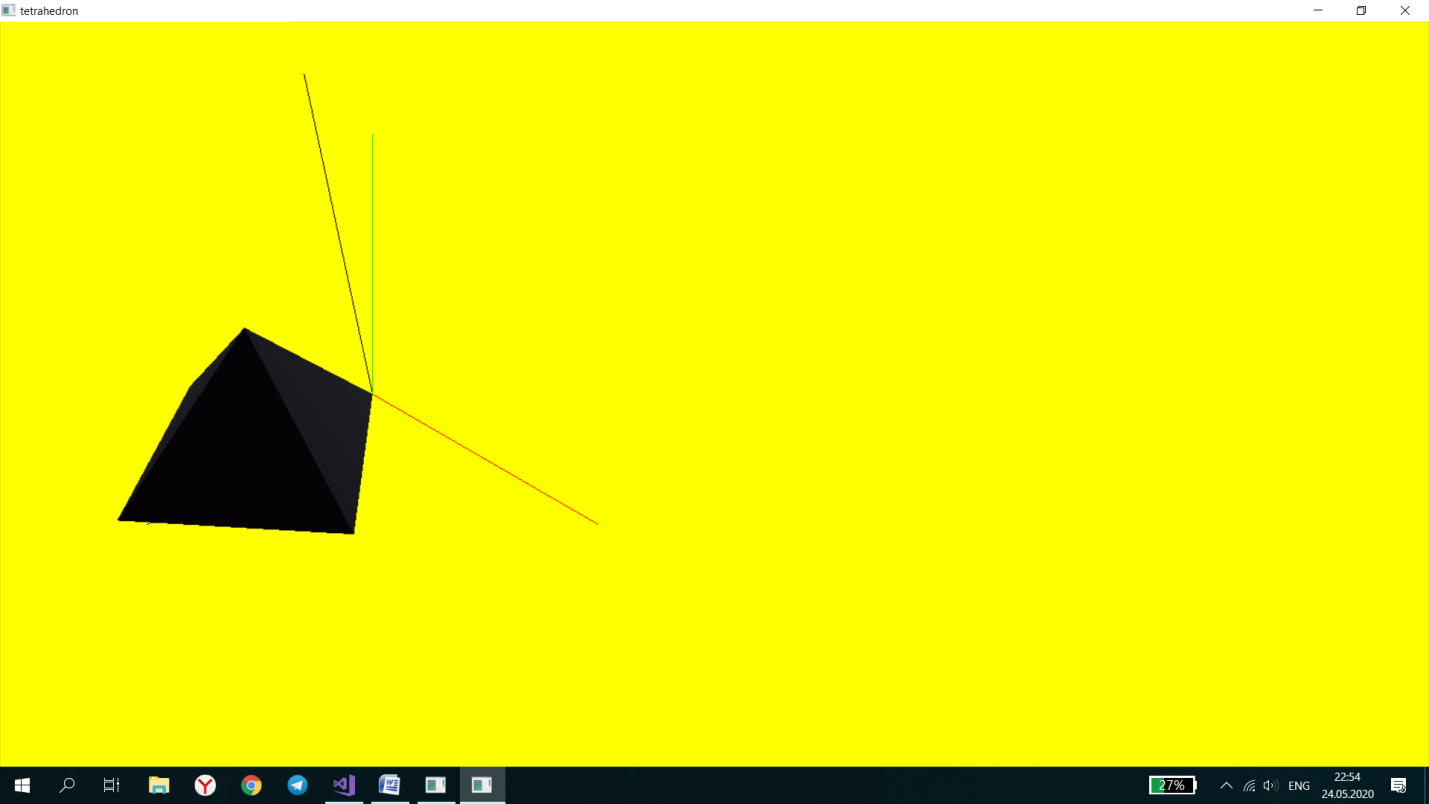
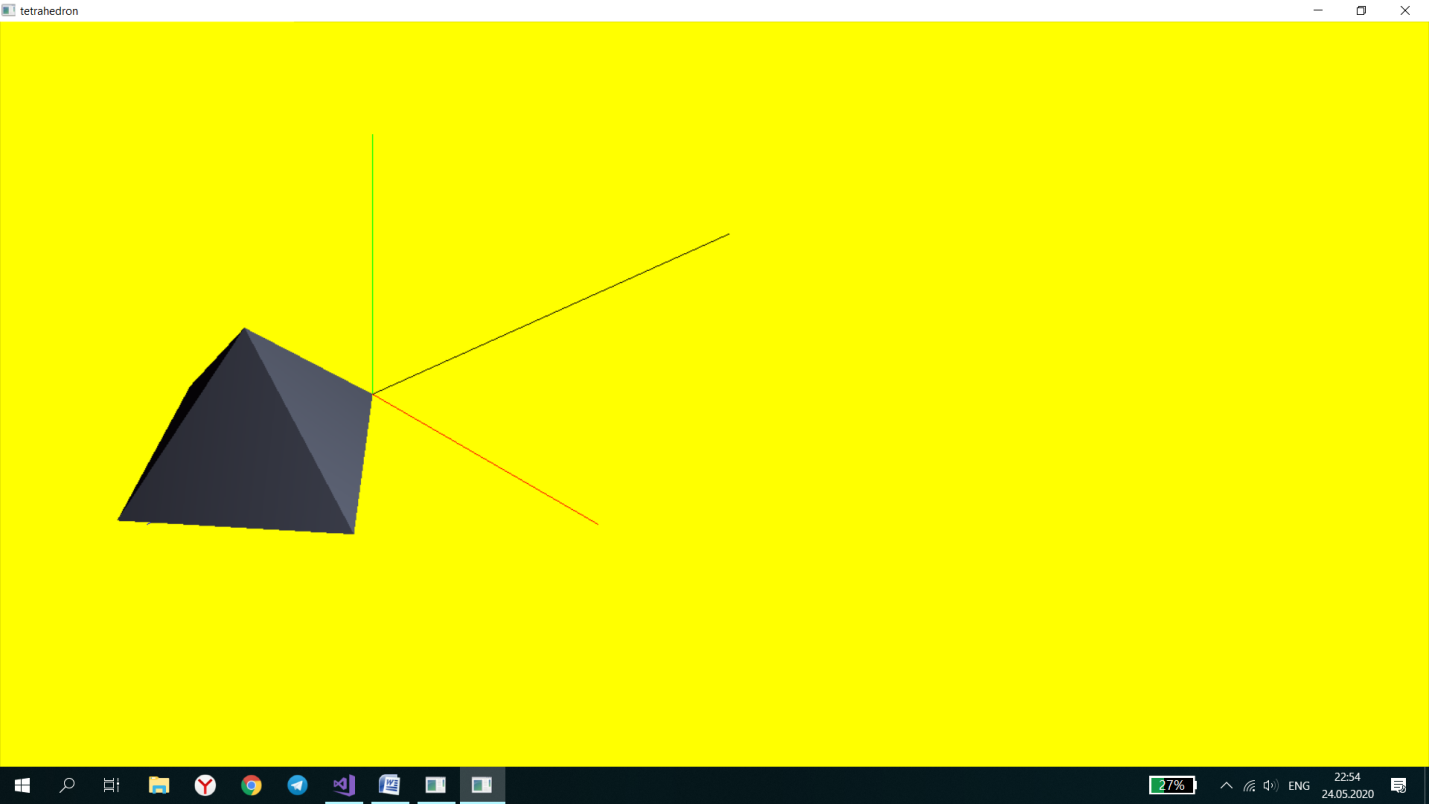
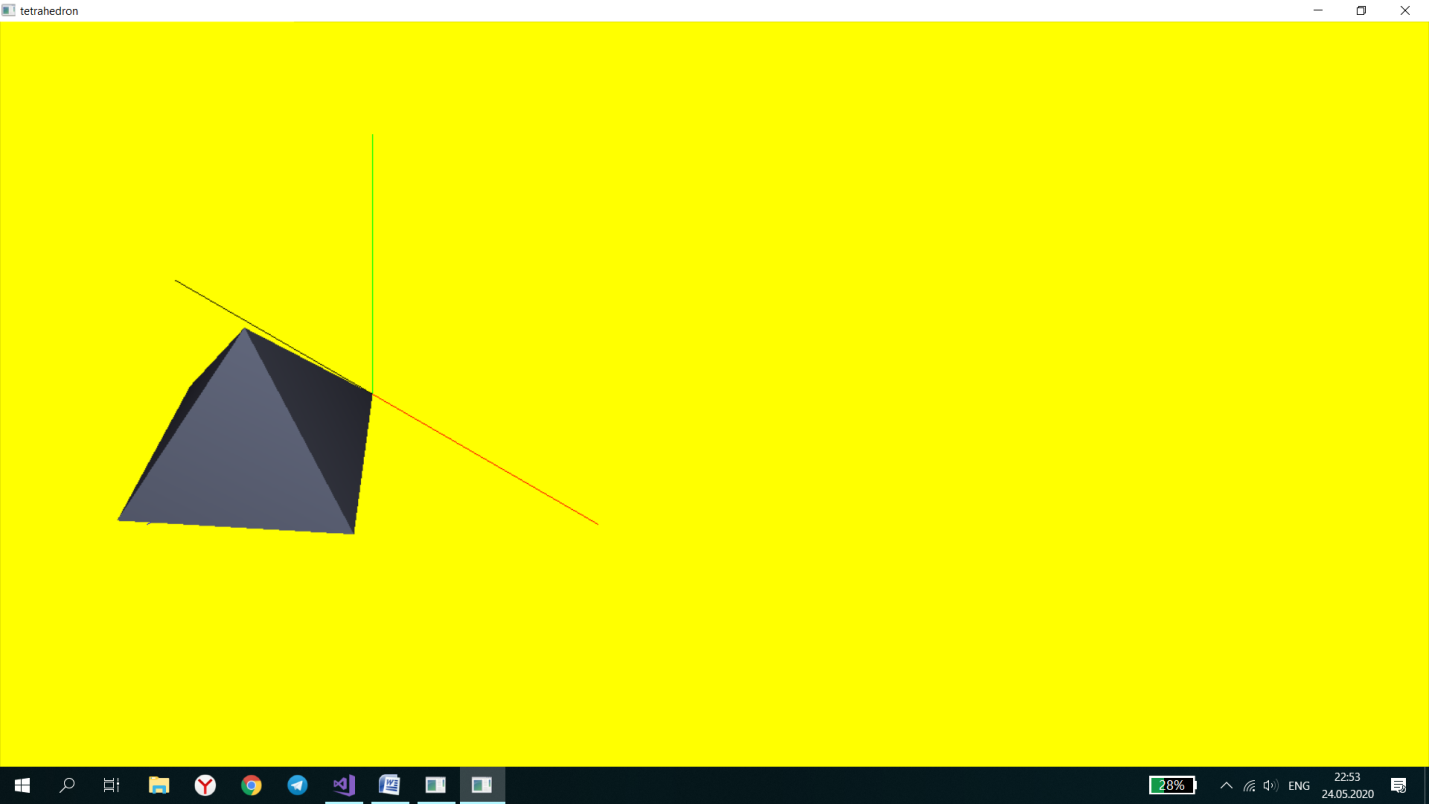
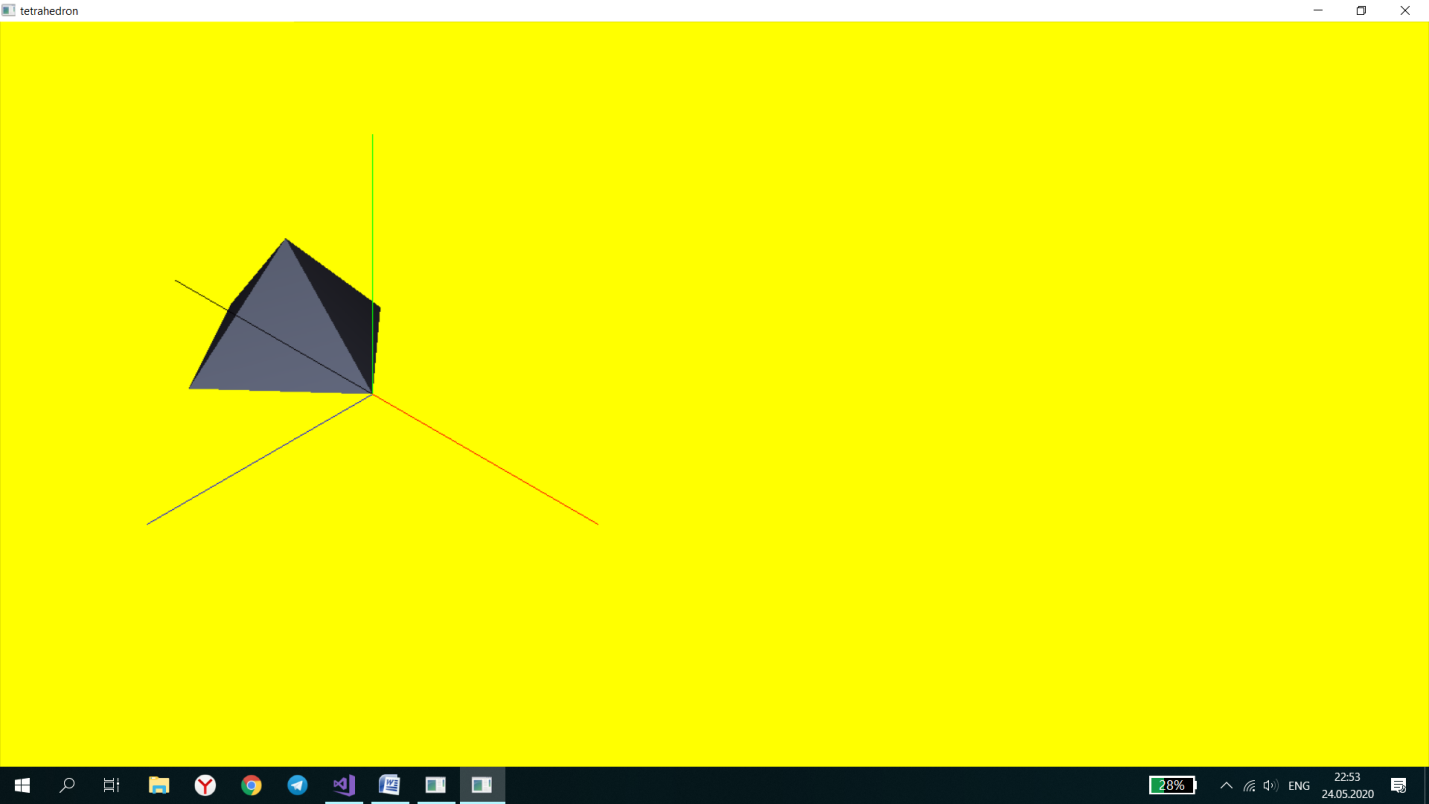
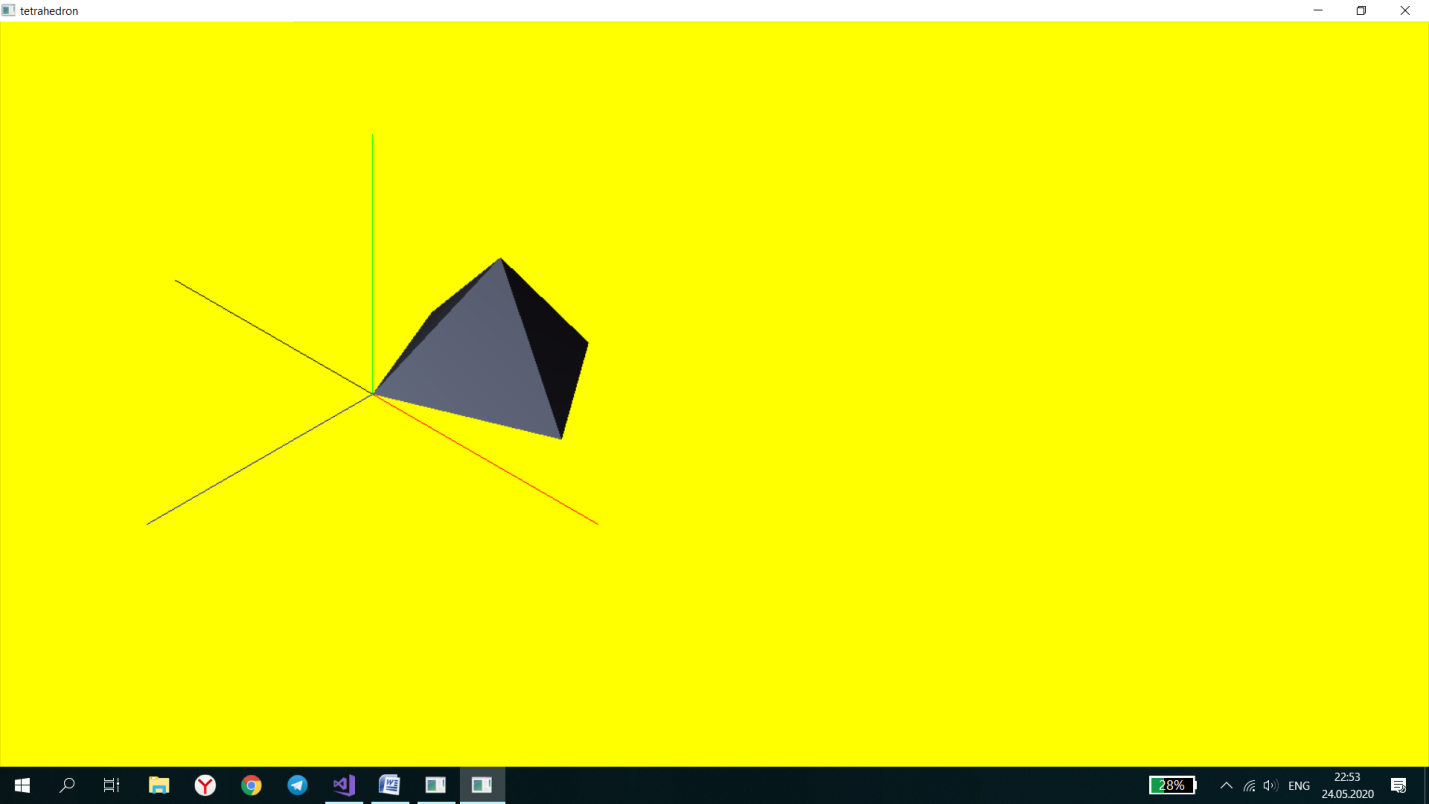
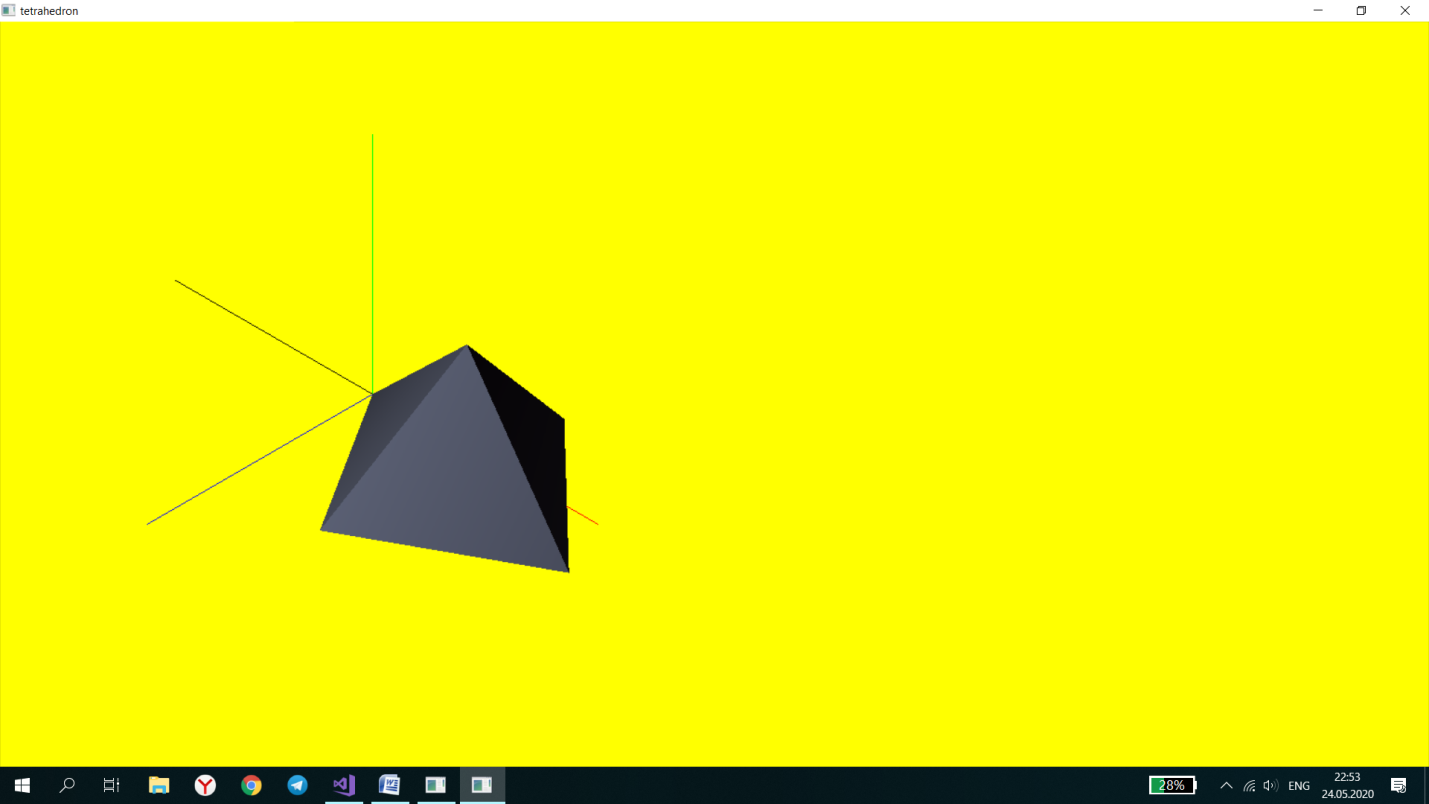
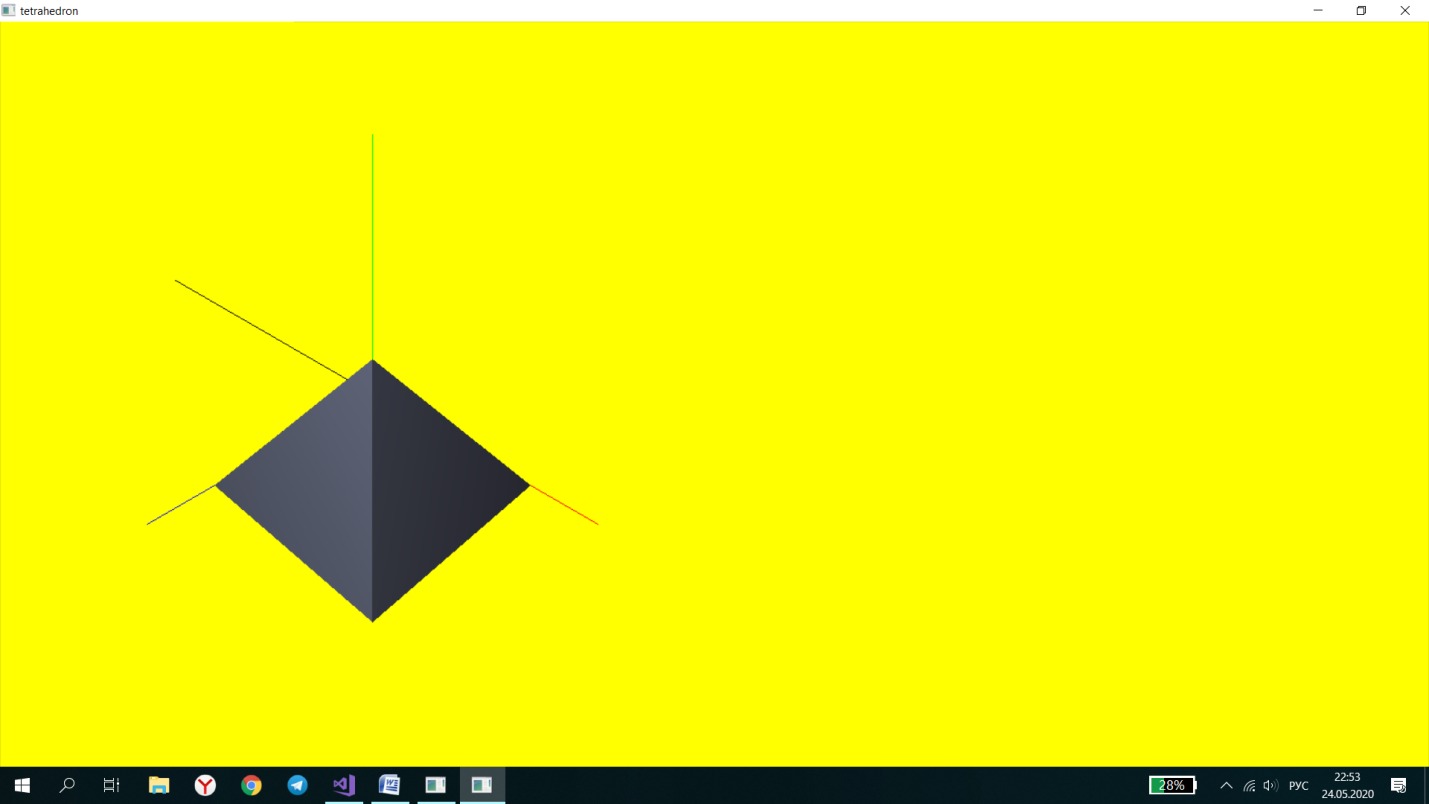
init();

glutMainLoop();

}

Результат работы.

Отображение лицевых граней при вращении:



Отображение нелицевых граней при вращении:

