**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 № 5

по дисциплине «Компьютерная графика»

**Выполнил**

студент группы 4211

Абдуллаев Л.Х.

Казань

2020

Цель :

1. Осуществите рисование объекта, созданного в предыдущей лабораторной работе, при помощи дисплейного списка.

2. Наложите на объект три различные текстуры на три различные грани. (Для этого необходимо предварительно создать текстурные изображения в формате \*.bmp.)

3. Осуществите изменение параметров отображения текстуры объекта таким образом, чтобы при нажатии на левую кнопку мыши объект излучал свет согласно заданной текстуре, при нажатии на правую кнопку мыши объект закрашивался текстурой, и при этом моделировались коэффициенты отражения.

Листинг:

#include <math.h>

#include <Gl\glut.h>

#include <C:\Users\lazzi\Desktop\программирование\C++ comp gf\GLAUX\GLAUX.H>

#pragma comment(lib, "C:\\Users\\lazzi\\Desktop\\программирование\\C++ comp gf\\GLAUX\\GLAUX.LIB ")

#pragma comment(lib, "legacy\_stdio\_definitions.lib")

GLfloat angle = 0;

GLfloat plusangle = 0;

GLfloat angle2 = 0;

GLfloat plusangle2 = 0;

int flagO = 1;

int flagL = 1;

AUX\_RGBImageRec\* image1 = auxDIBImageLoadA("C:\\Users\\lazzi\\Desktop\\программирование\\C++ comp gf\\GLAUX\\tx1.bmp");

AUX\_RGBImageRec\* image2 = auxDIBImageLoadA("C:\\Users\\lazzi\\Desktop\\программирование\\C++ comp gf\\GLAUX\\tx2.bmp");

AUX\_RGBImageRec\* image3 = auxDIBImageLoadA("C:\\Users\\lazzi\\Desktop\\программирование\\C++ comp gf\\GLAUX\\tx3.bmp");

GLuint list = 0;

void init(void)

{

glClearColor(1.0, 1.0, 1.0, 0.0);

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluPerspective(60, 1, 1, 10);

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

list = glGenLists(1);

glNewList(list, GL\_COMPILE);

glBegin(GL\_POLYGON);

glNormal3f(0, -1, 0);

glVertex3f(1.5, 0, 1.5);

glVertex3f(1.5, 0, 0);

glVertex3f(0, 0, 0);

glVertex3f(0, 0, 1.5);

glEnd();

glBegin(GL\_POLYGON);

glNormal3f(0, 0.557, -0.557);

glVertex3f(0.75, 1, 0.75);

glVertex3f(0, 0, 0);

glVertex3f(1.5, 0, 0);

glEnd();

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, image1->sizeX, image1->sizeY, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, image1->data);

glEnable(GL\_TEXTURE\_2D);

glEnable(GL\_DEPTH\_TEST);

glBegin(GL\_POLYGON);

glNormal3f(0, 0.557, 0.557);

glTexCoord2f(0, 0); glVertex3f(1.5, 0, 1.5);

glTexCoord2f(1, 0); glVertex3f(0, 0, 1.5);

glTexCoord2f(0, 1); glVertex3f(0.75, 1, 0.75);

glEnd();

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, image2->sizeX, image2->sizeY, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, image2->data);

glEnable(GL\_TEXTURE\_2D);

glEnable(GL\_DEPTH\_TEST);

glBegin(GL\_POLYGON);

glNormal3f(-0.557, 0.557, 0);

glTexCoord2f(0, 0); glVertex3f(0, 0, 1.5);

glTexCoord2f(1, 0); glVertex3f(0, 0, 0);

glTexCoord2f(0, 1); glVertex3f(0.75, 1, 0.75);

glEnd();

glTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGB, image3->sizeX, image3->sizeY, 0, GL\_RGB, GL\_UNSIGNED\_BYTE, image3->data);

glEnable(GL\_TEXTURE\_2D);

glEnable(GL\_DEPTH\_TEST);

glBegin(GL\_POLYGON);

glNormal3f(0.557, 0.557, 0);

glTexCoord2f(0, 0); glVertex3f(0.75, 1, 0.75);

glTexCoord2f(1, 0); glVertex3f(1.5, 0, 0);

glTexCoord2f(0, 1); glVertex3f(1.5, 0, 1.5);

glEnd();

glEndList();

}

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);

glPushMatrix();

glRotatef(angle2, 0, 1, 0);

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

glEnable(GL\_LIGHTING);

glEnable(GL\_LIGHT0);

glBegin(GL\_LINES);

glColor3f(1, 1, 1);

glVertex3f(1, 2, 2);

glVertex3f(0.0, 0.0, 0.0);

glEnd();

glPopMatrix();

glPushMatrix();

glRotatef(angle, 0, 1, 0);

glCallList(list);

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);

}

}

void mouseButton(int button, int state, int x, int y){

if (button == GLUT\_LEFT\_BUTTON) glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_DECAL);

else if (button == GLUT\_RIGHT\_BUTTON) glTexEnvi(GL\_TEXTURE\_ENV, GL\_TEXTURE\_ENV\_MODE, GL\_MODULATE);

}

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

{

glutInit(&argc, argv);

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

glutInitWindowPosition(0, 0);

glutCreateWindow("lab5");

glutDisplayFunc(myDisplay);

glutKeyboardFunc(keys);

glutReshapeFunc(myReshape);

glutIdleFunc(myIdle);

glutMouseFunc(mouseButton);

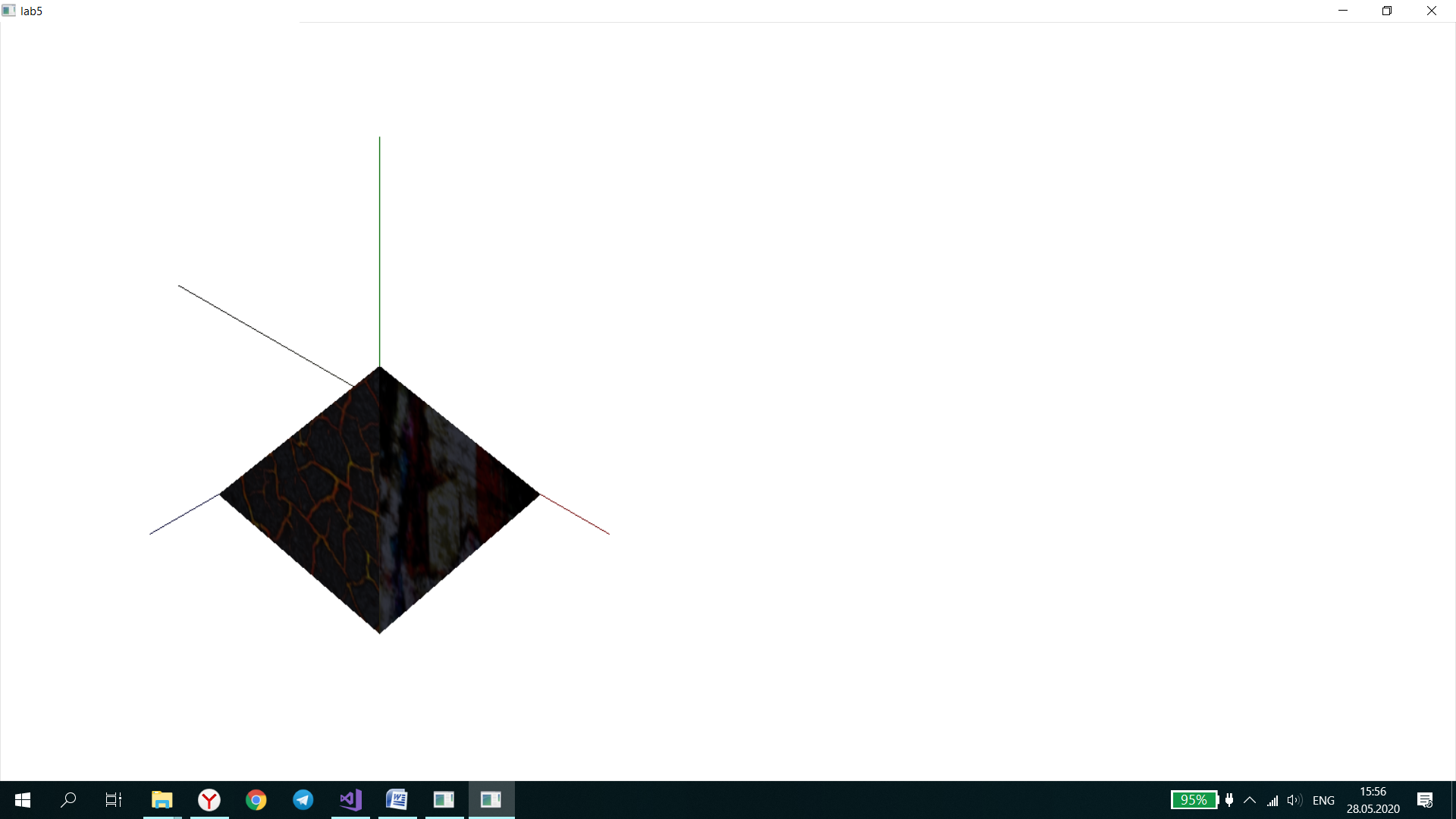
init();

glutMainLoop();

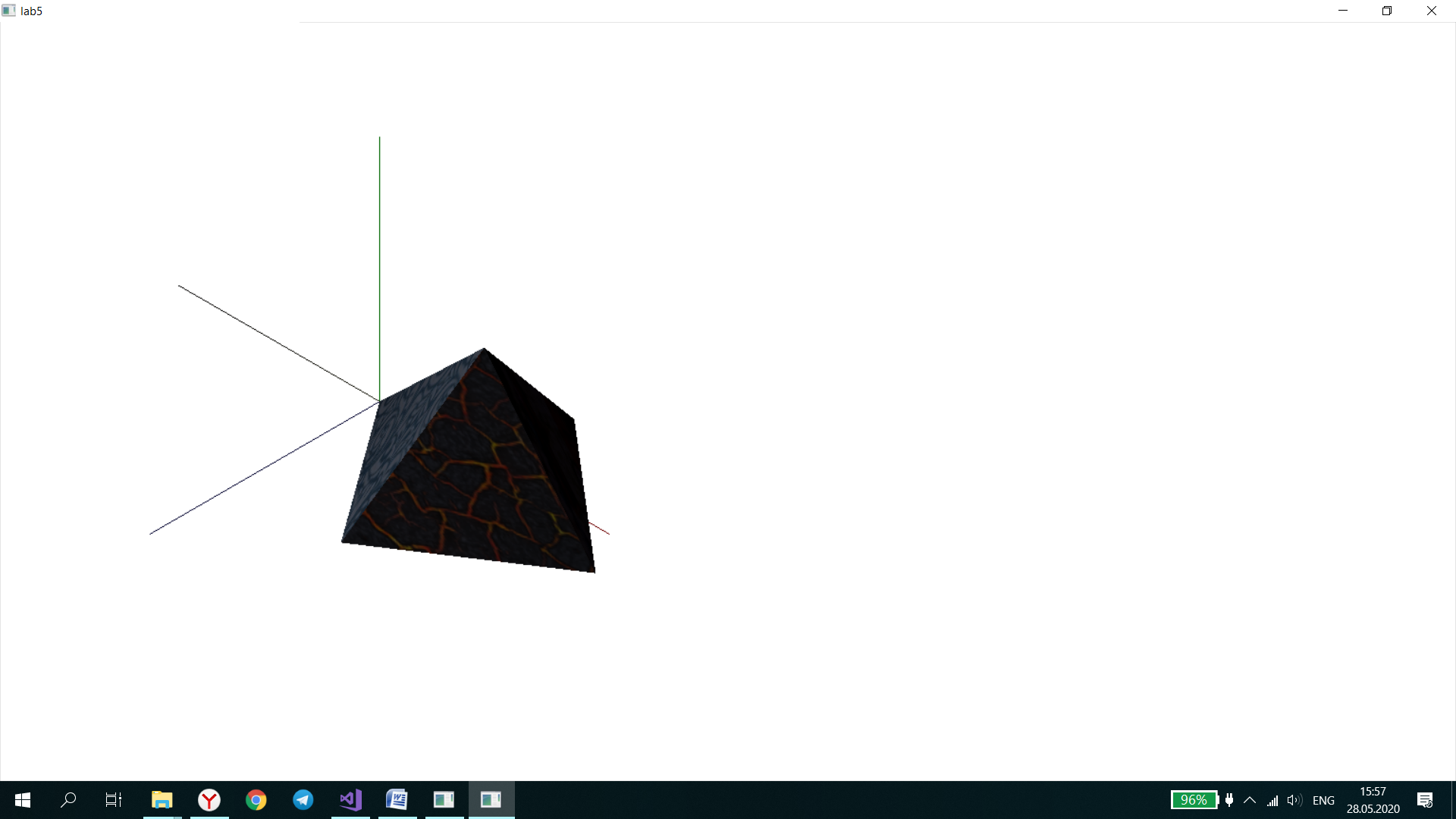
}

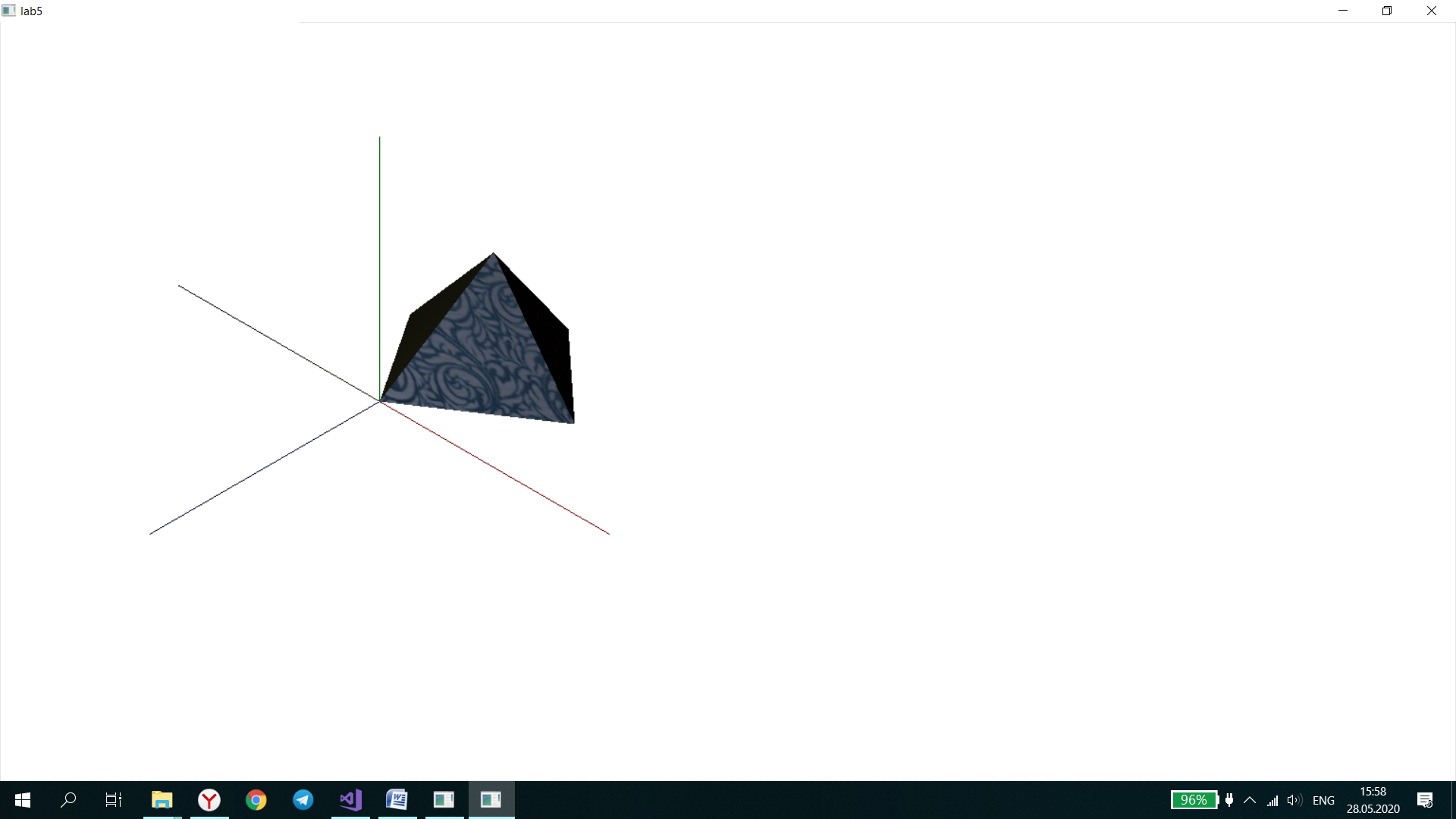
Результат работы:

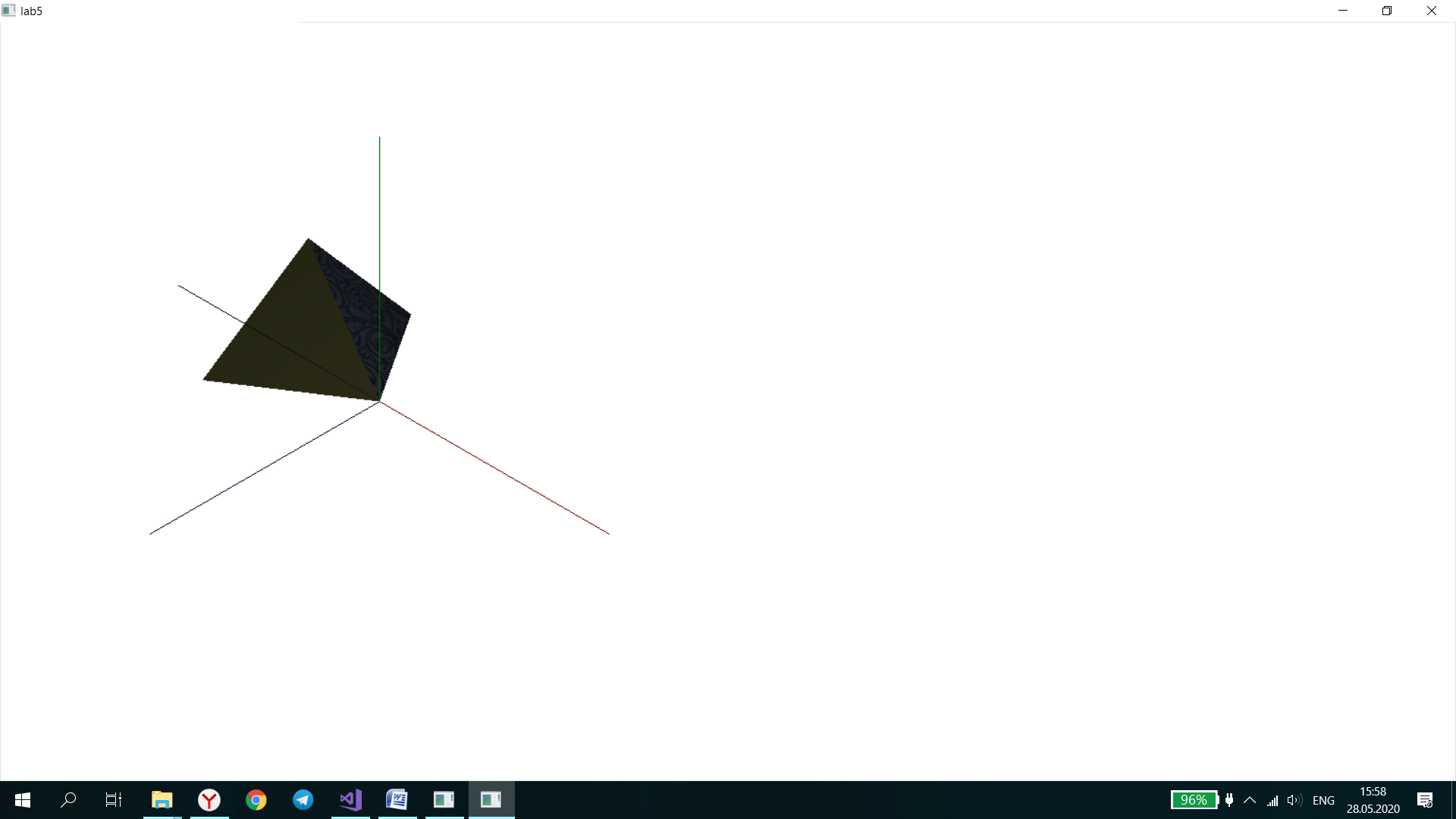
Начальное положение

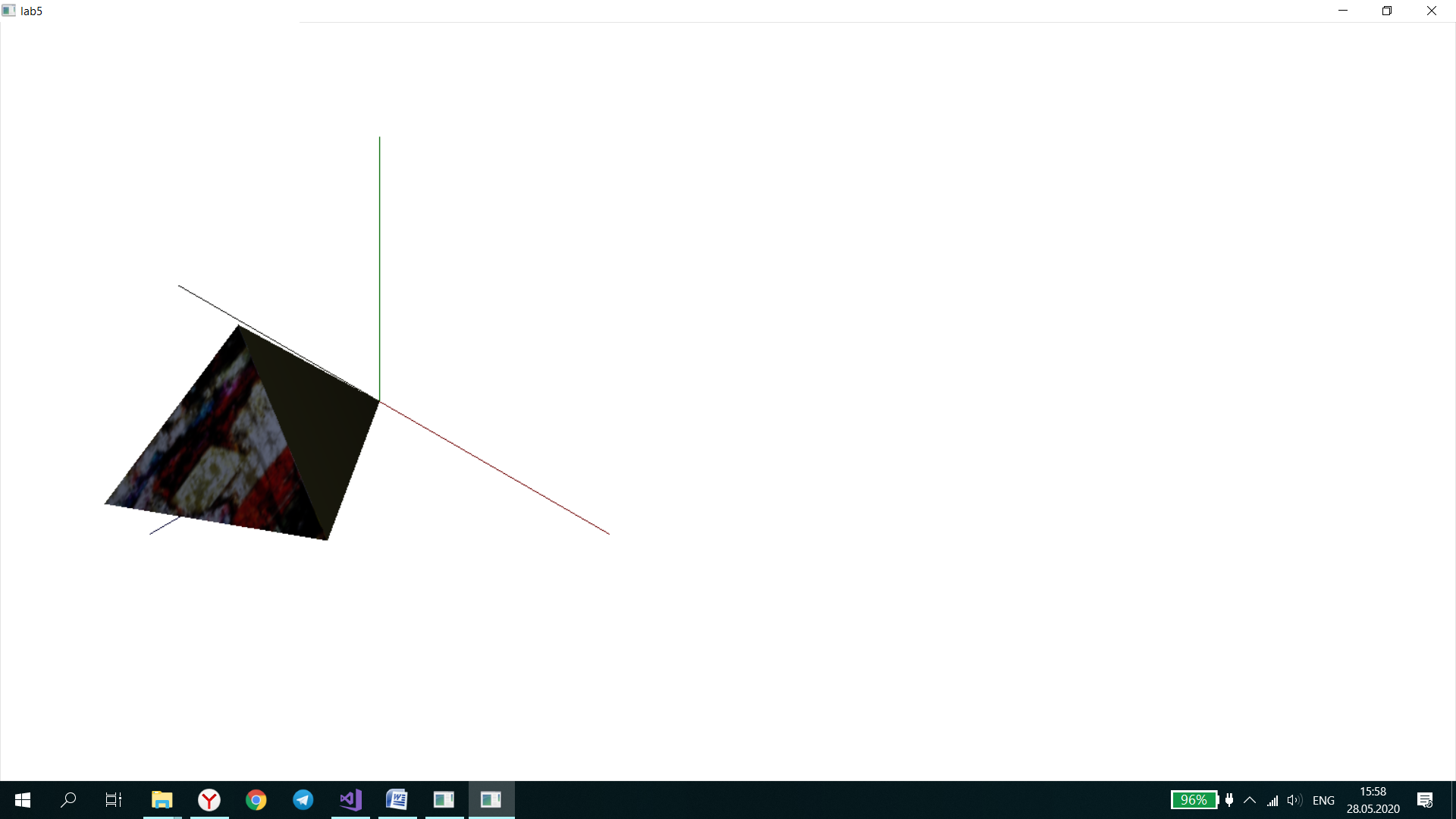


Движение объекта

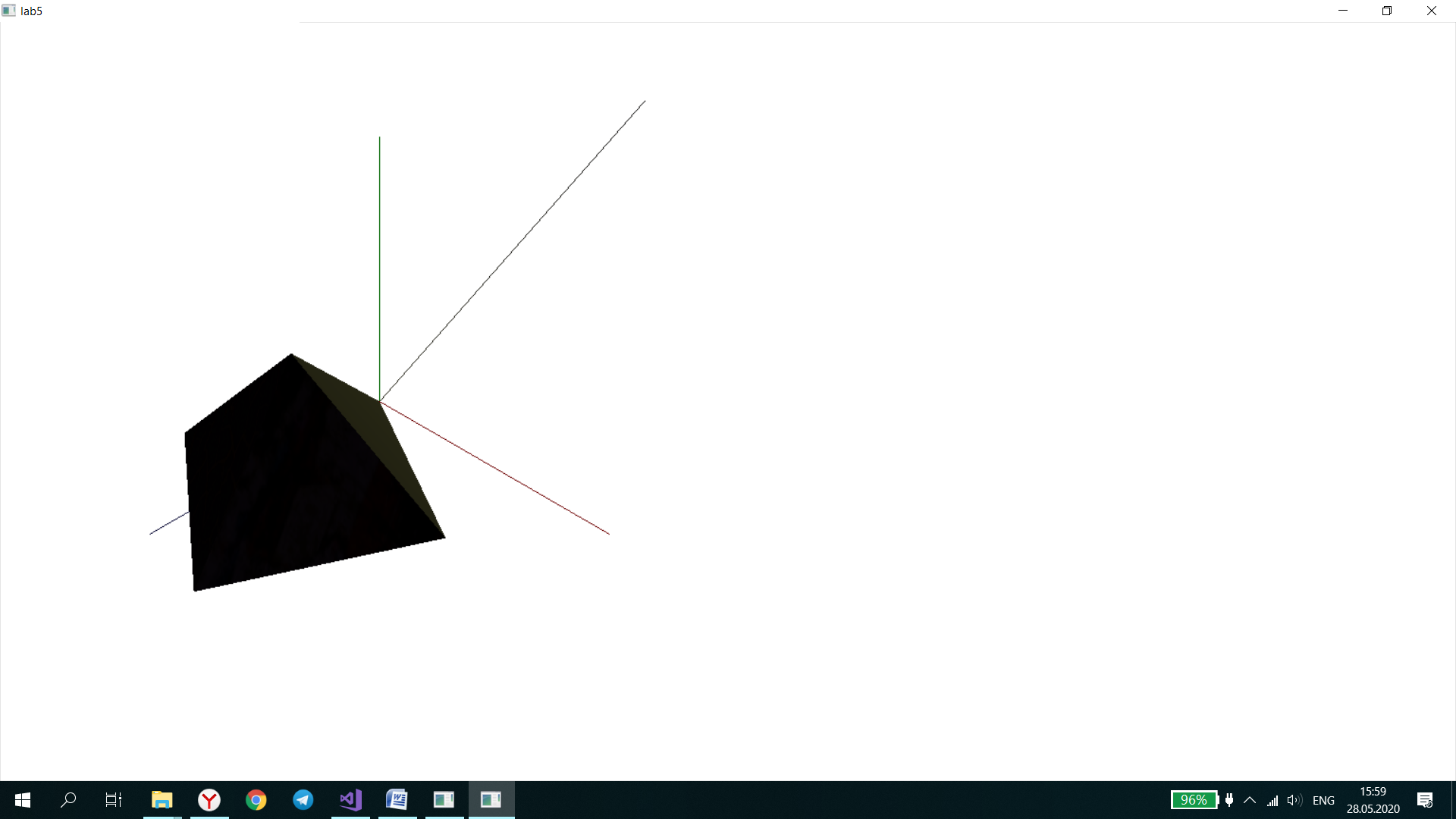


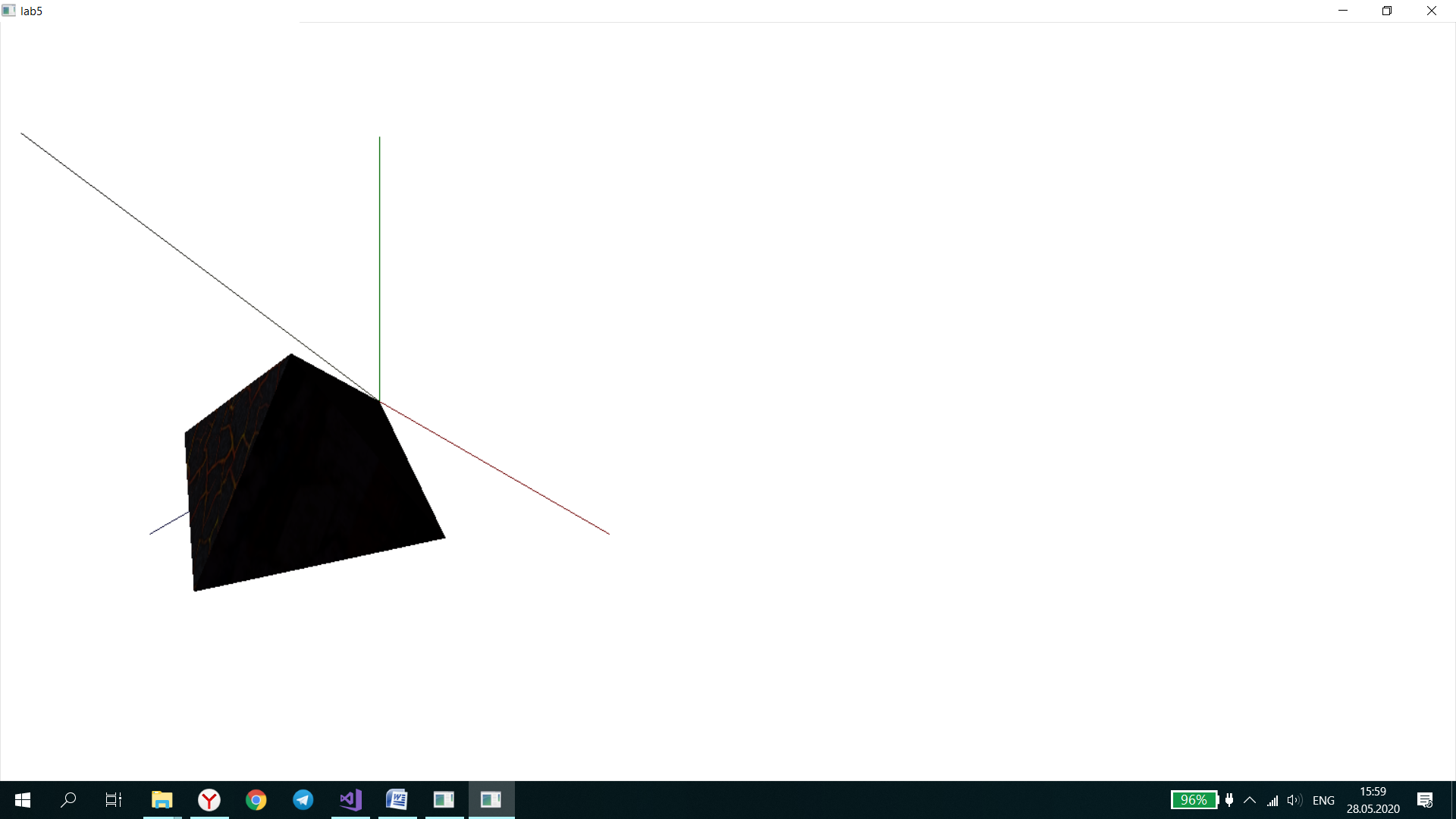


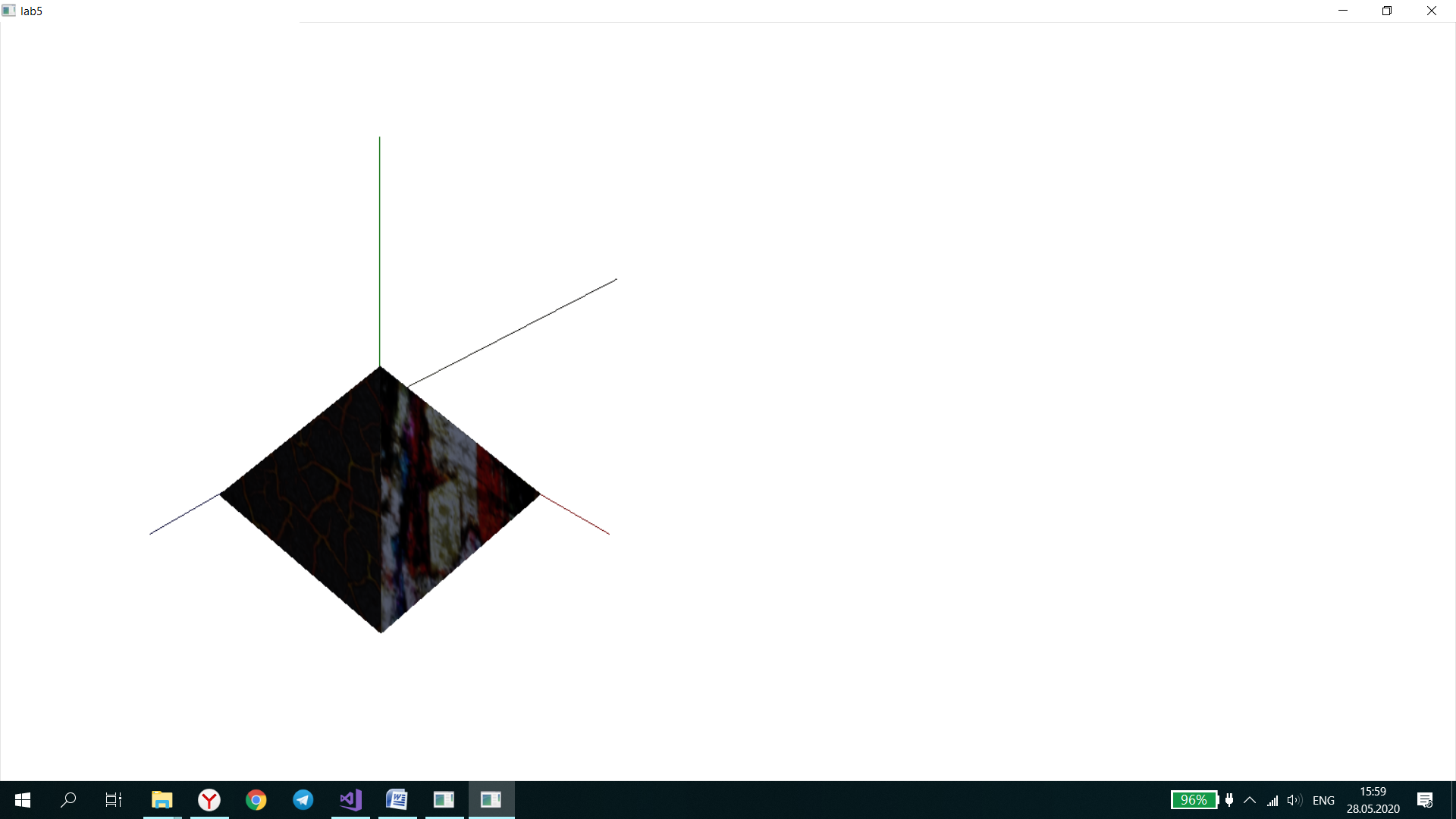




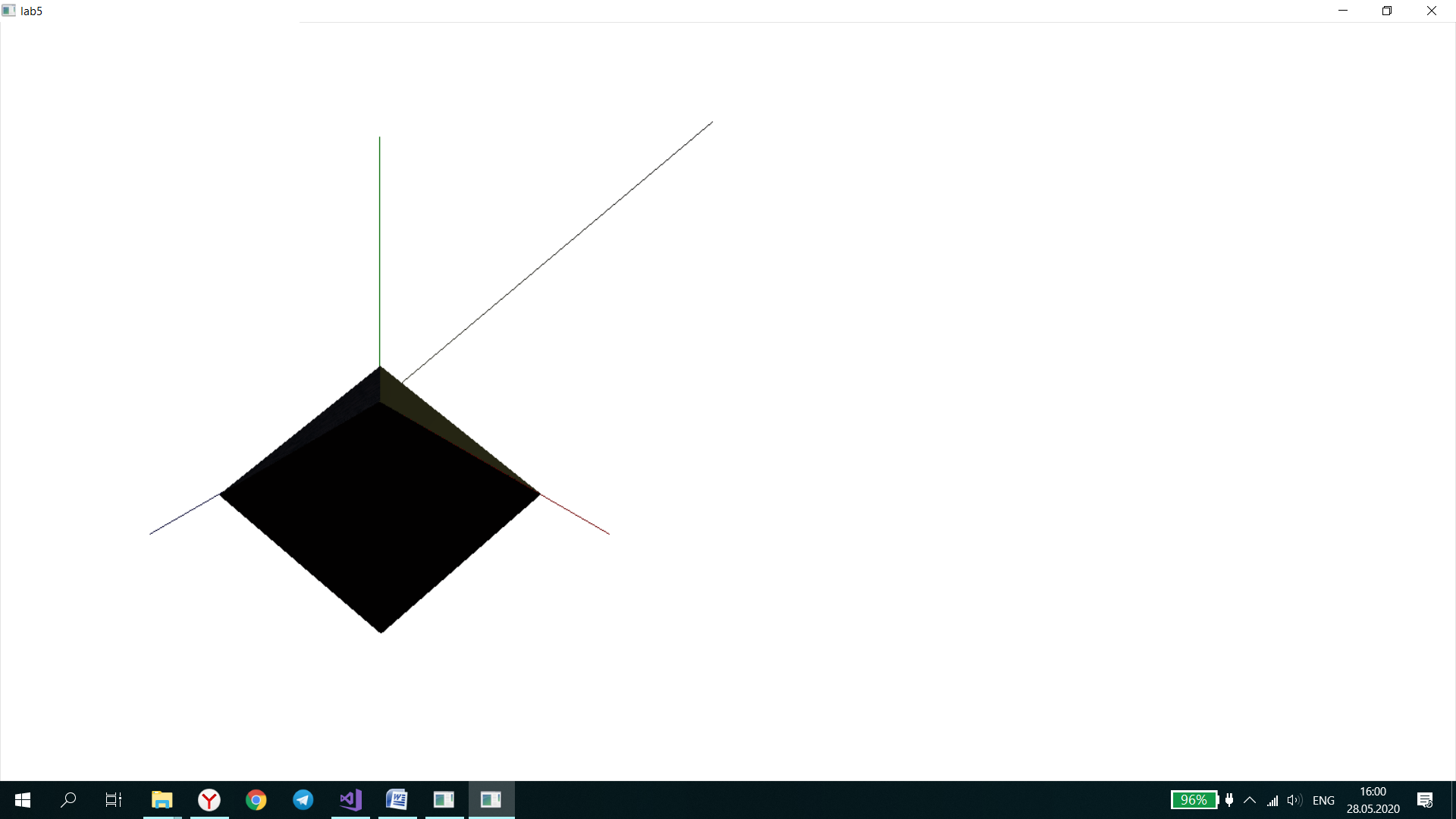
Движение источника света

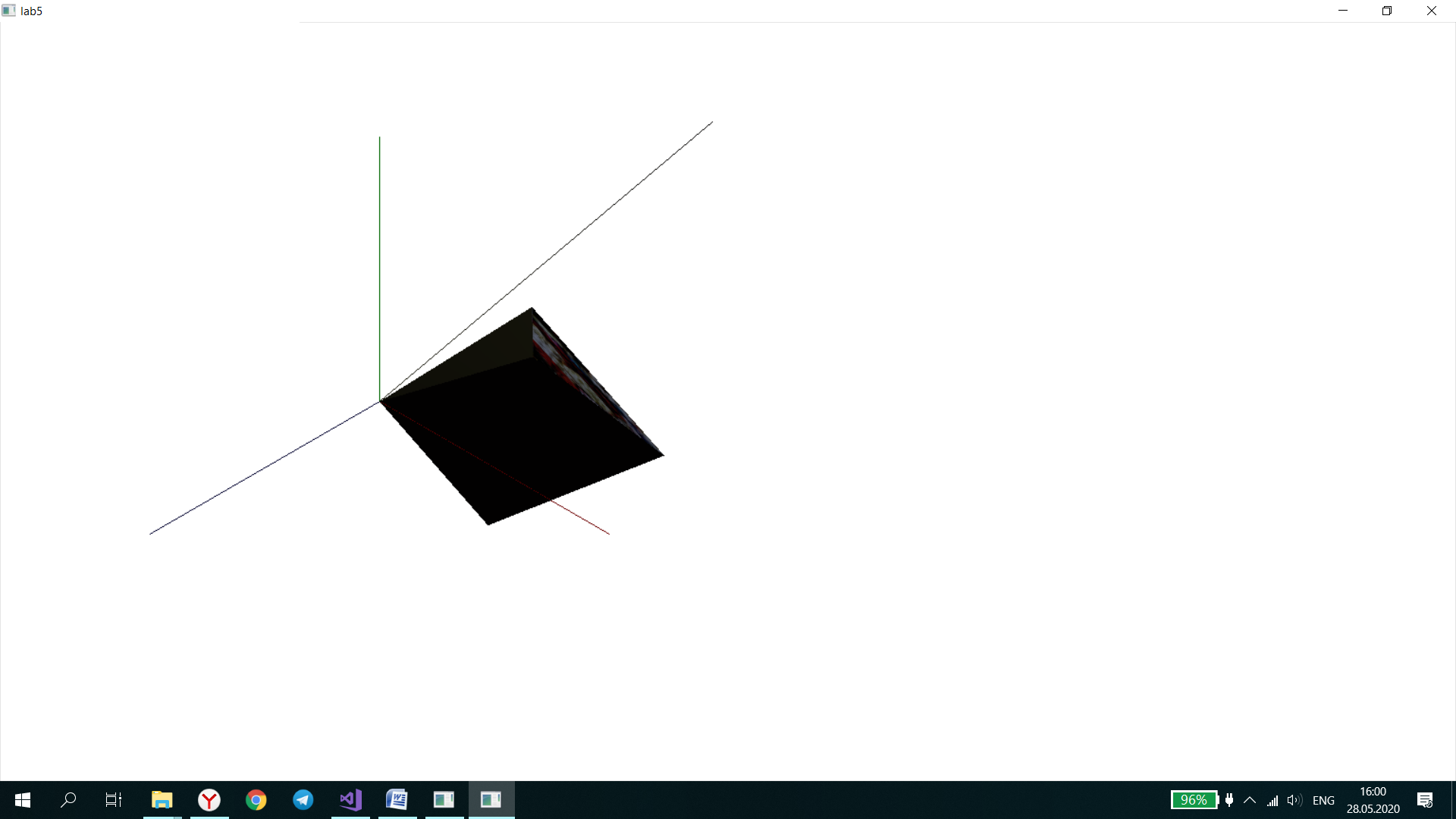


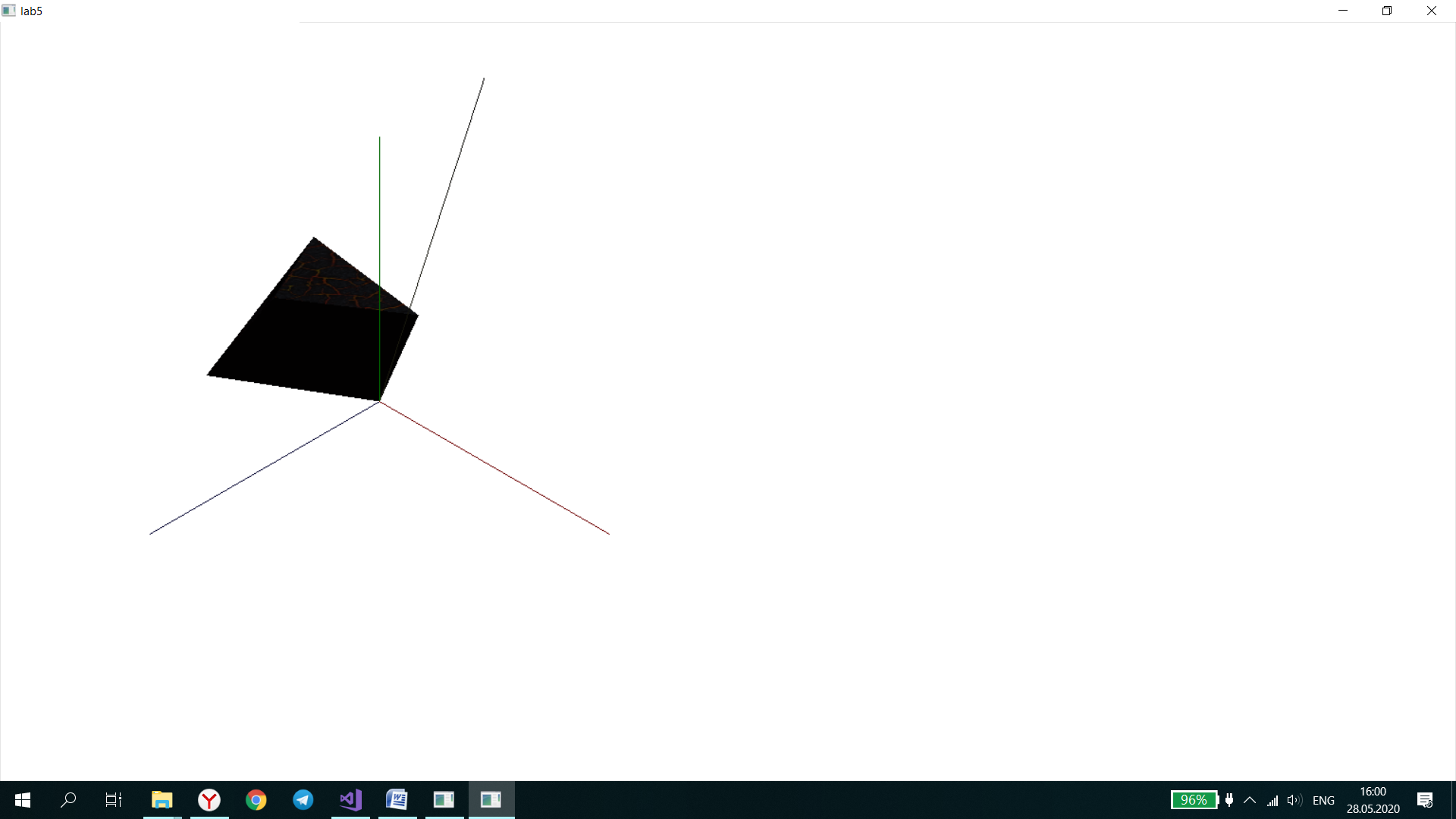


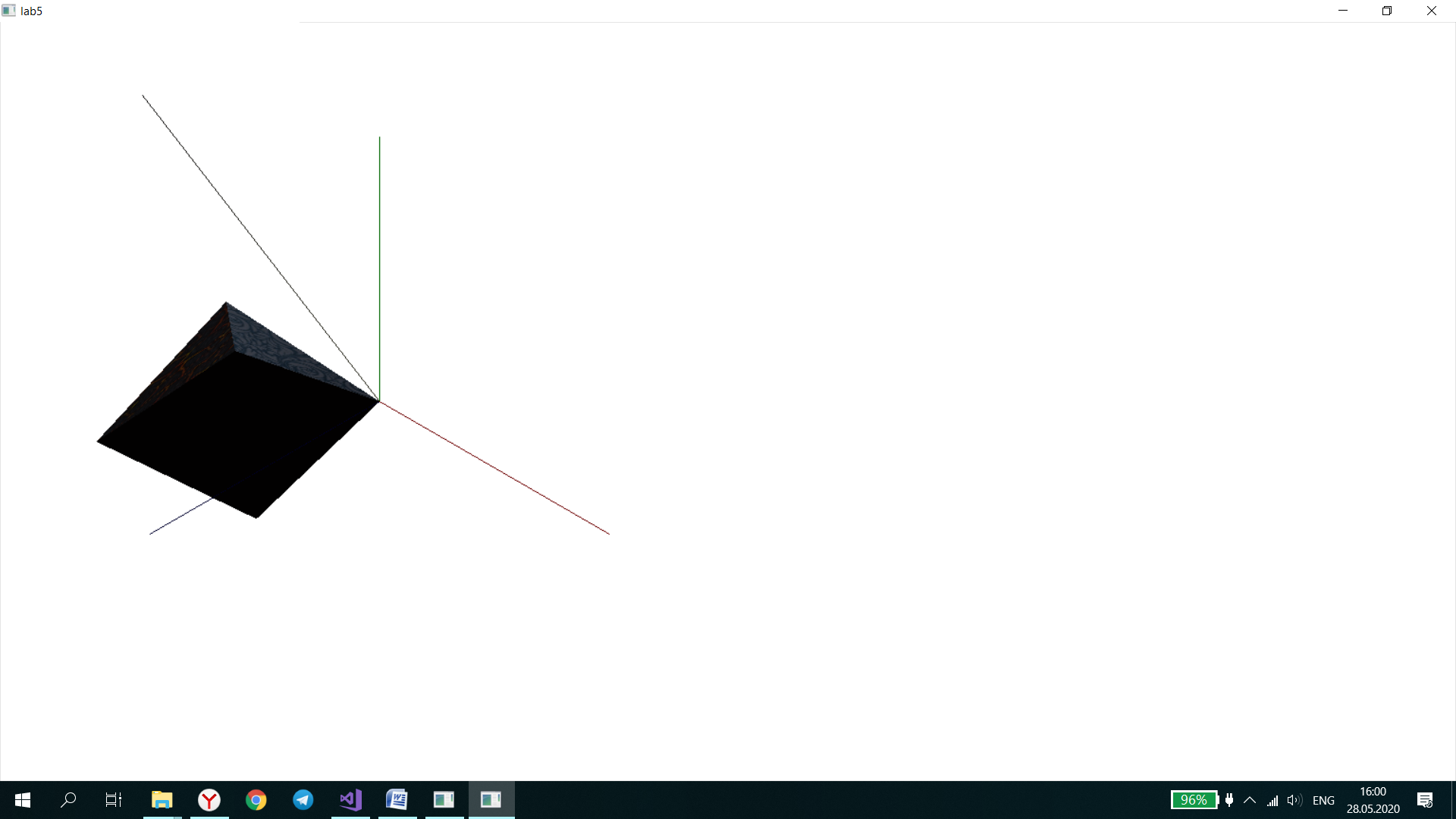


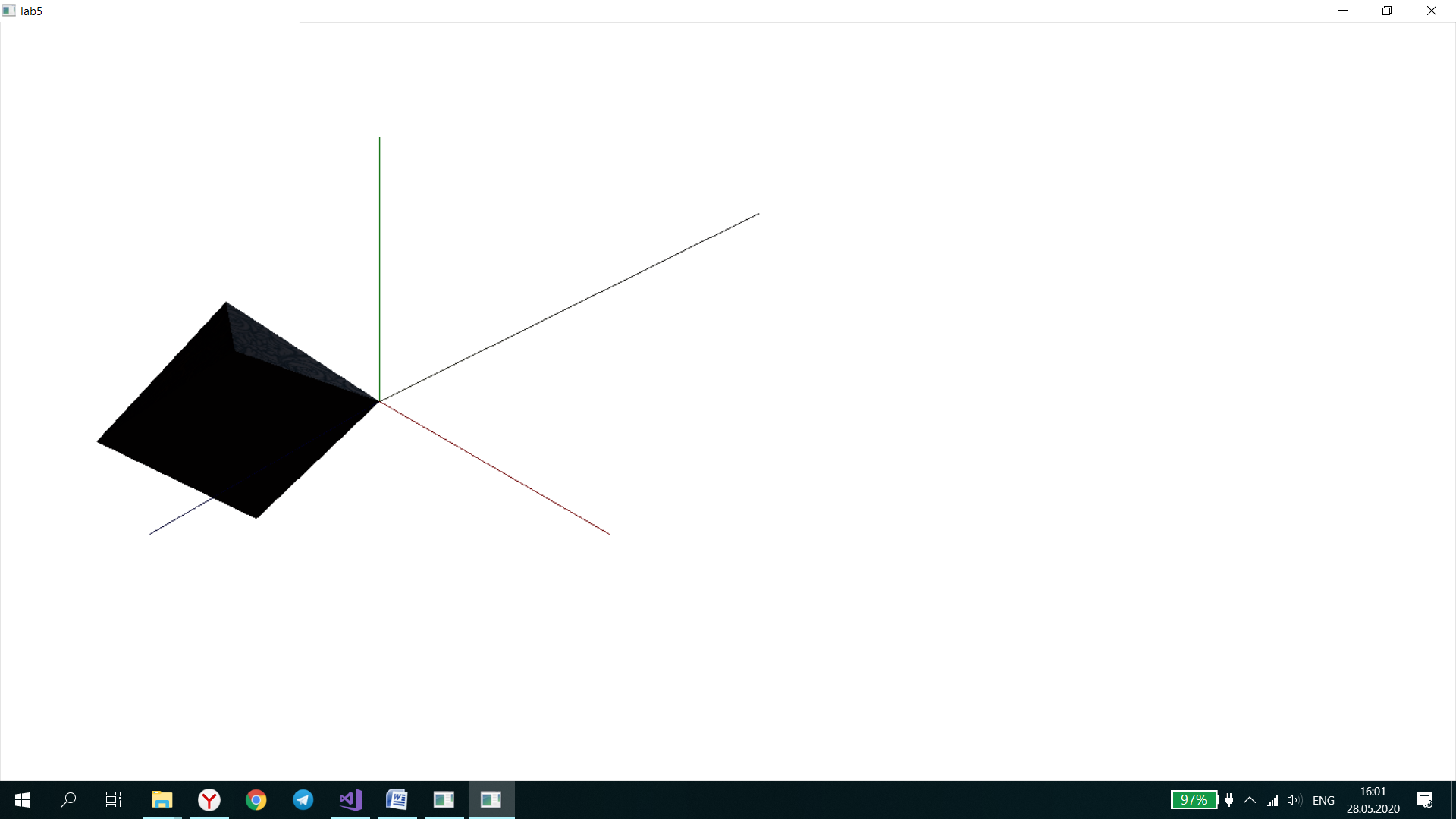
Показ нелицевых граней



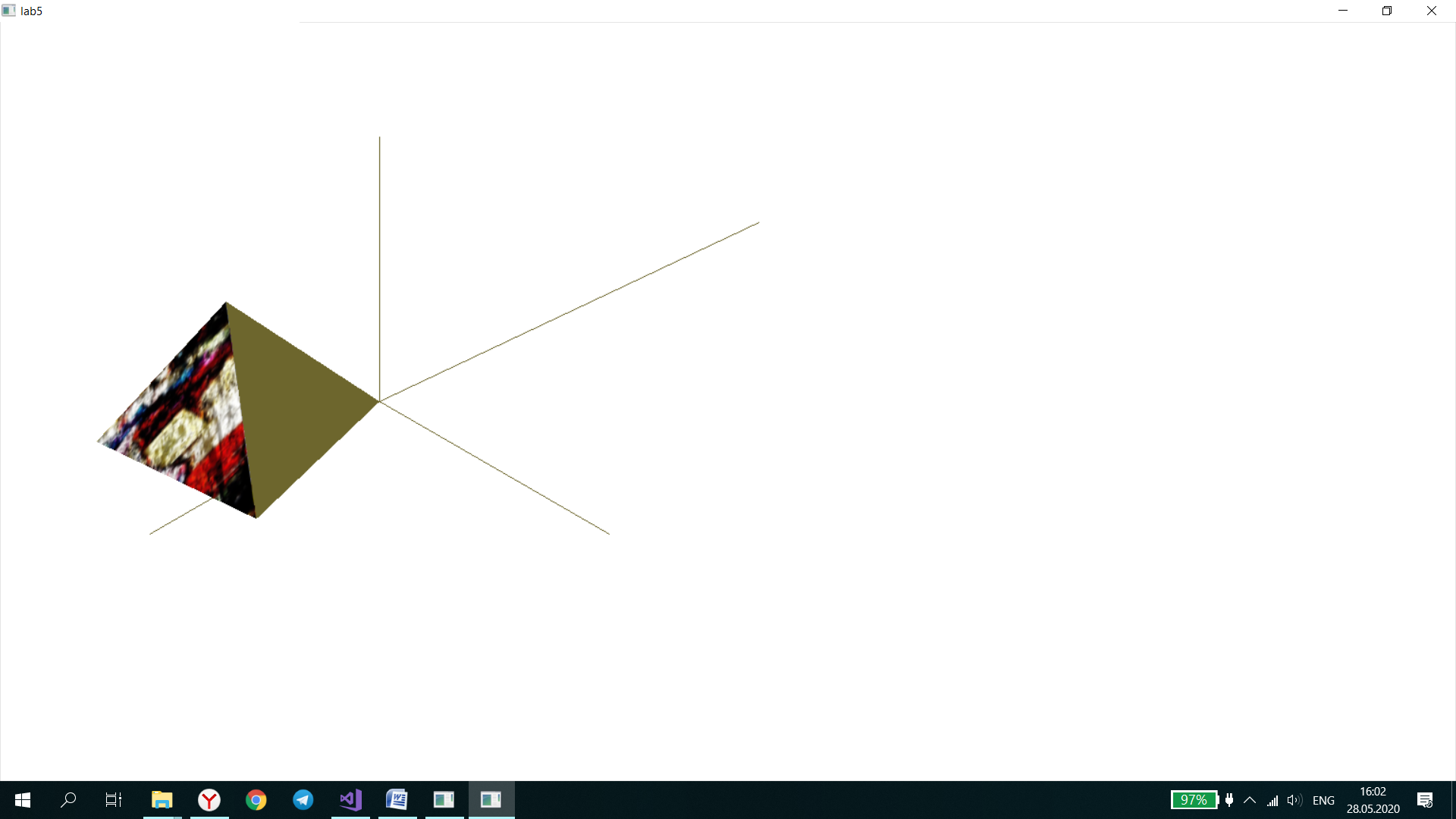


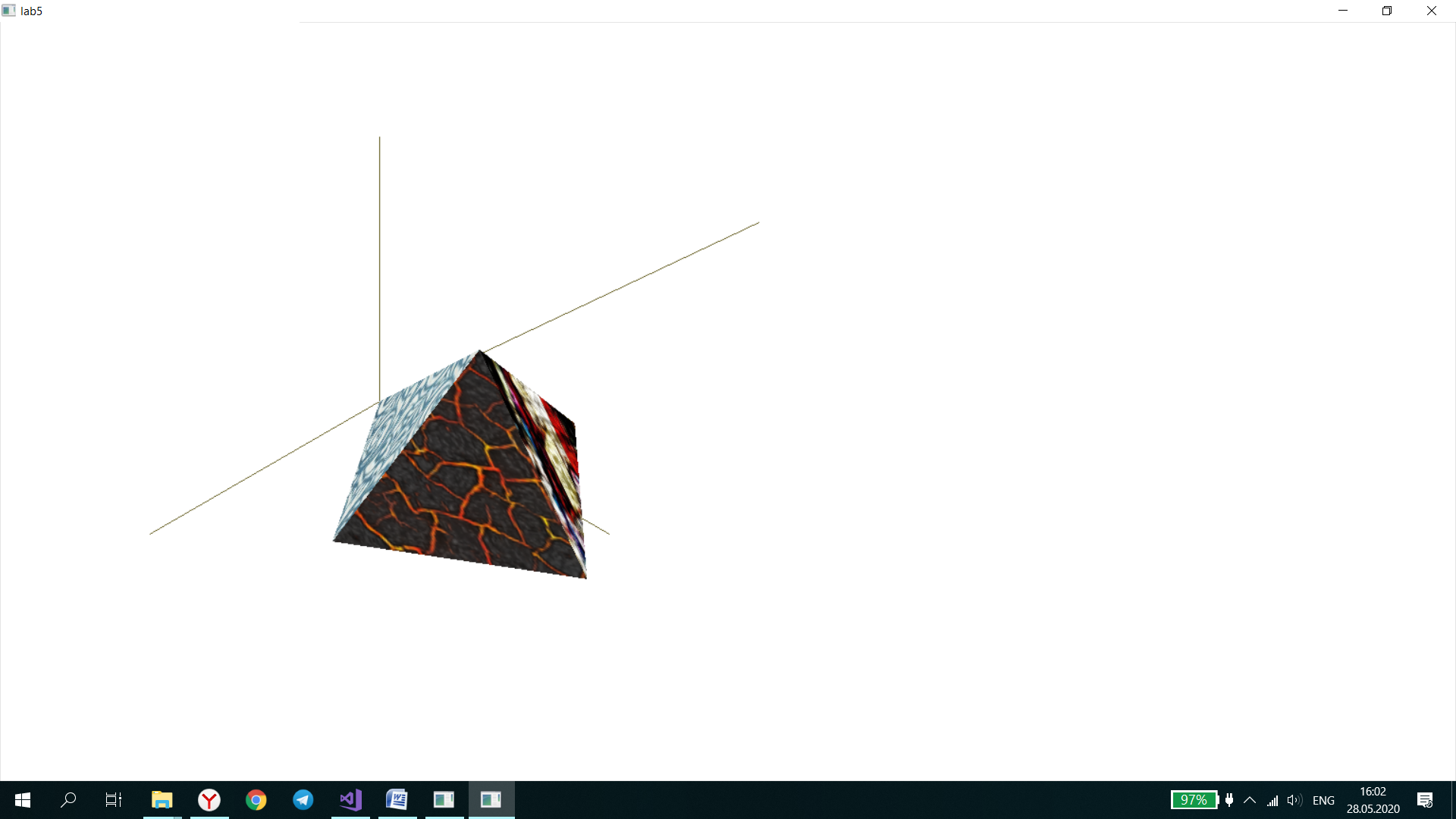


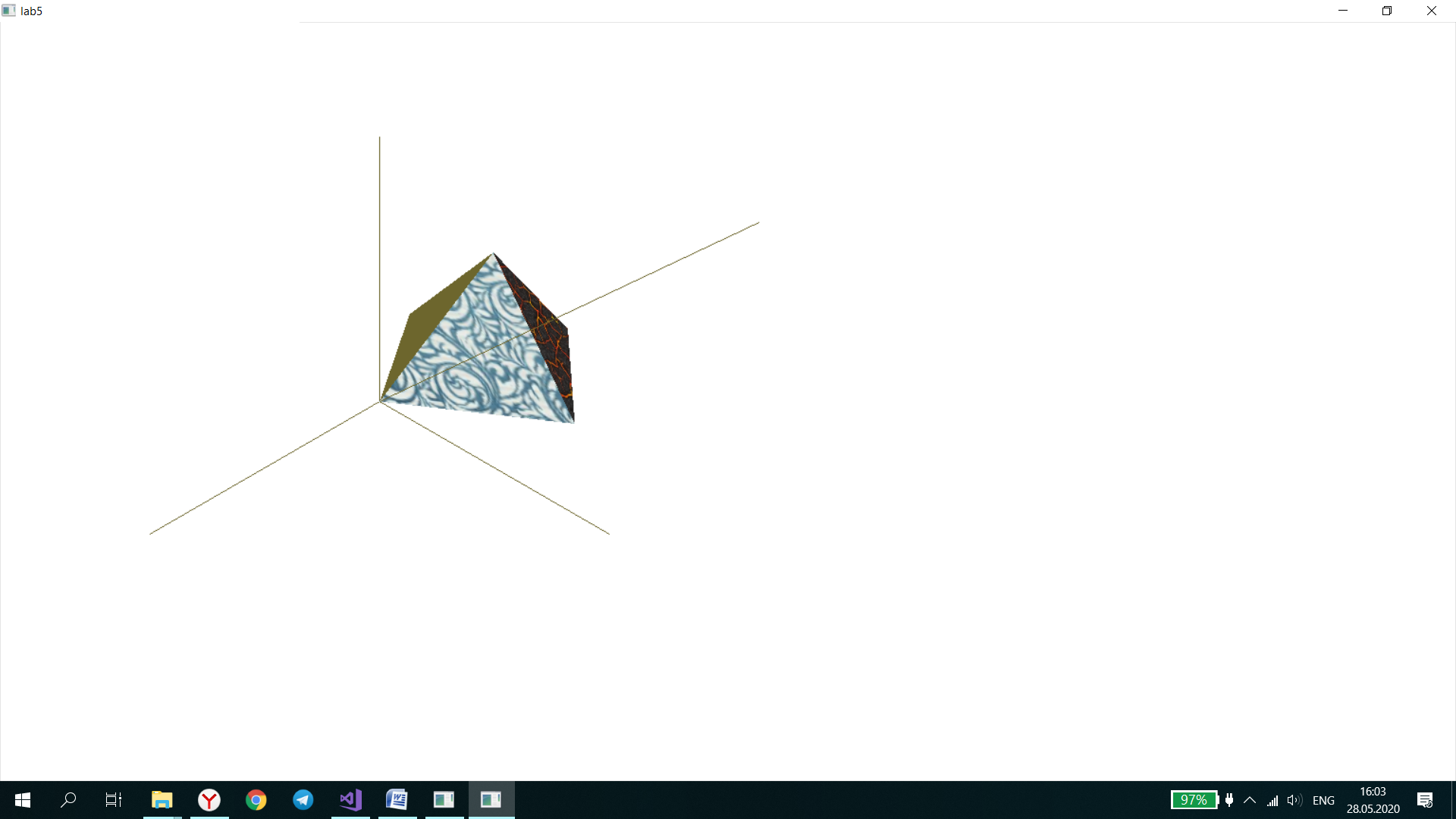


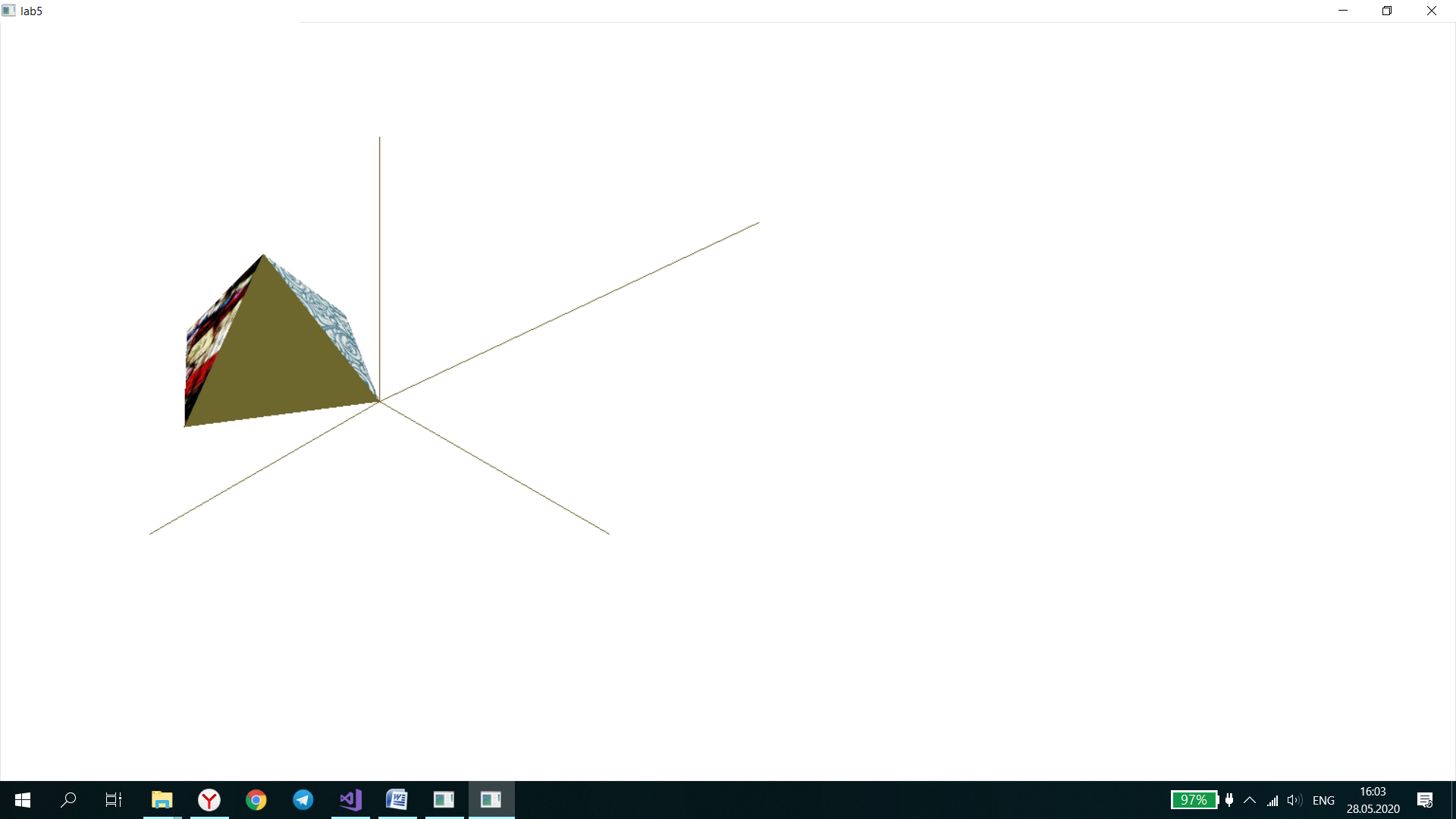


Нажатие на левую кнопку мыши (лицевые грани)









Нажатие на левую кнопку мыши (нелицевые грани)

