|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Кафедра |  | O7 |  | Информационные системы и программная инженерия |
|  |  | шифр |  | наименование кафедры, по которой выполняется работа |
| Дисциплина |  | Компьютерная геометрия и графика | | |
|  |  | наименование дисциплины | | |

|  |  |  |
| --- | --- | --- |
| ЛАБОРАТОРНАЯ РАБОТА | 4 |  |
|  | номер (при наличии) |  |
| Трехмерные построения. | | |
| Буфер глубины. Видовые параметры. | | |
| Параллельная и перспективная проекции. | | |

при наличии указать тему лабораторной работы и (или) номер варианта

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ОБУЧАЮЩИЙСЯ** | | | | | | |
| группы | | | |  | | И913Б |
|  |  | Кульга Н.И | | | | |
| подпись |  | фамилия и инициалы | | | | |
|  | | | | |
| дата сдачи | | | | |
| **ПРОВЕРИЛ** | | | | | | |
|  | | | | | | | |
| ученая степень, ученое звание, должность | | | | | | | |
|  |  | Мажайцев Е.А | | | | | |
| подпись |  | фамилия и инициалы | | | | | |
| Оценка / балльная оценка | | |  | | | |
|  | | | | |
| дата проверки | | | | |

Задание 1.

Постройте изображения трех плоских фигур в пространстве (в соответствии с вариантом). Оцените полученный результат: правильно ли изображены фигуры, если нет - объясните почему.

Задание 2.

Установите режим проверки буфера глубины в программе к заданию 1.

Сравните полученный результат с предыдущим.

Задание 3.

Установите видовые параметры в соответствии с вариантом и параллельную проекцию; постройте изображение куба с координатами вершин, равными по модулю 1 (куб изображается с помощью 12 треугольников). Оцените результат.

Задание 4.

Установите перспективную проекцию и постройте каркасное изображение куба.

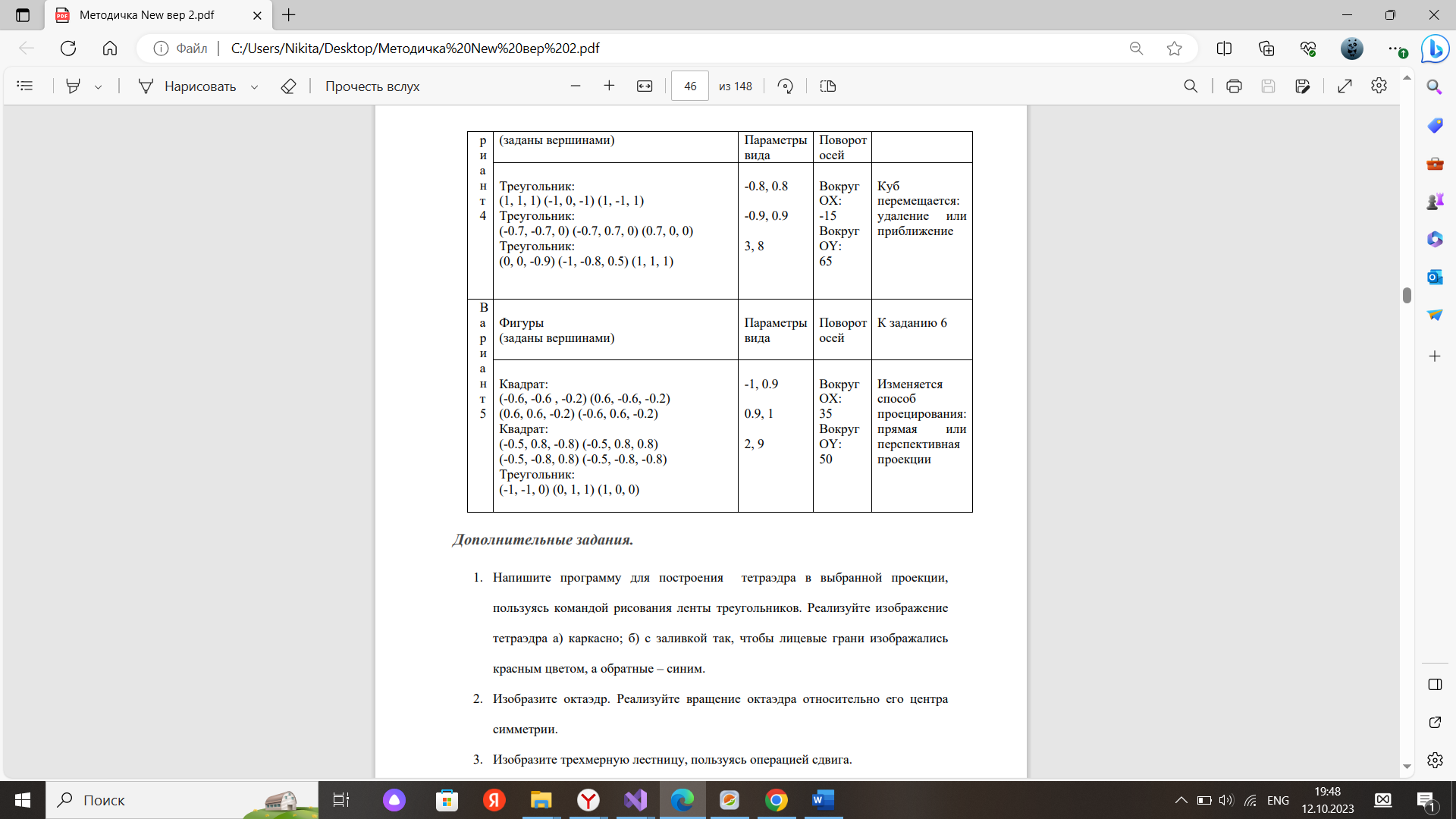
Задание 5.

Поверните оси координат в соответствии с вариантом и постройте изображение куба. Оцените результат. Почему части куба отсечены?

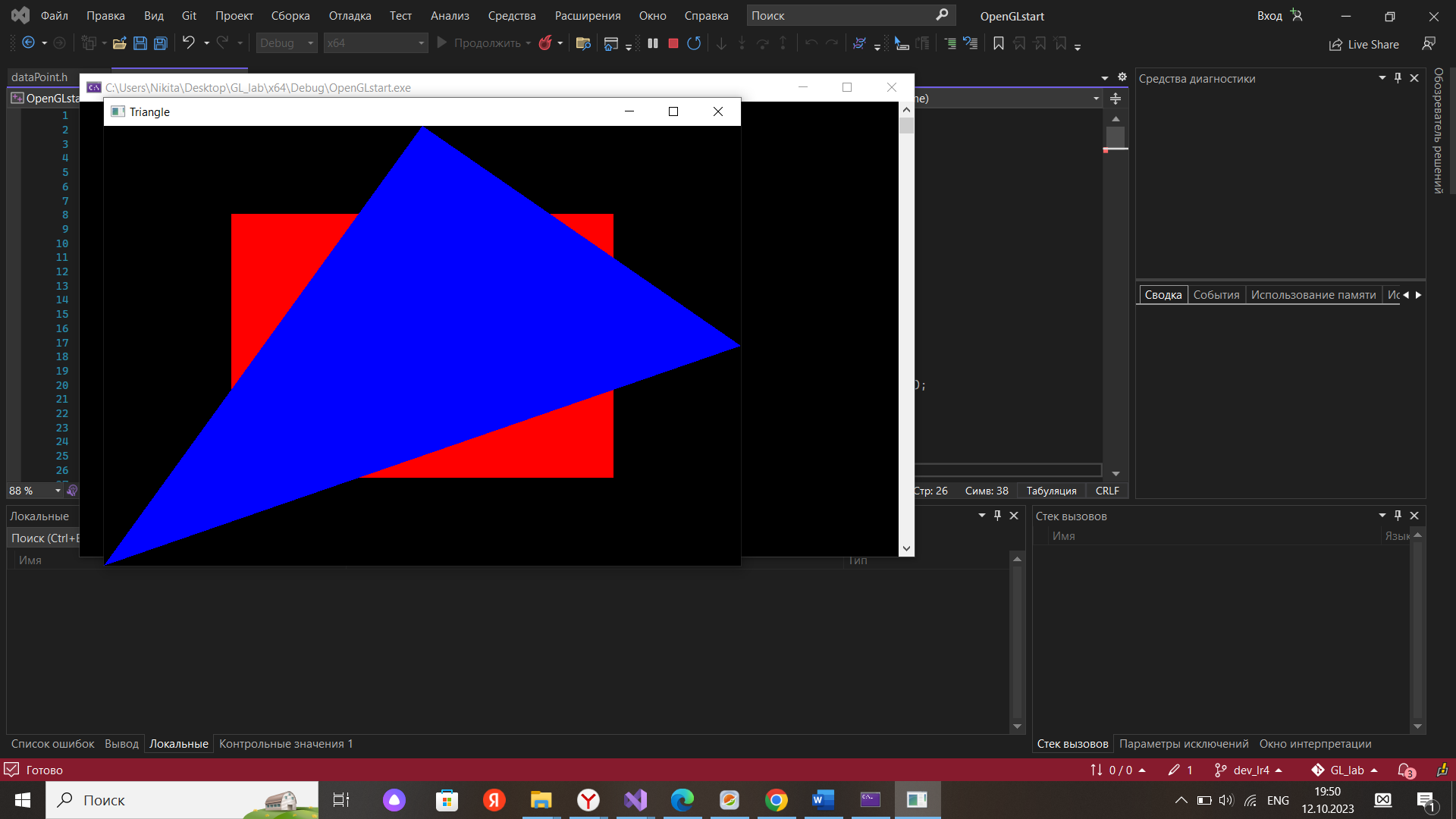
Измените видовые параметры таким образом, чтобы куб изображался без отсечений. Измените код программы таким образом, чтобы каждая грань куба изображалась своим цветом.

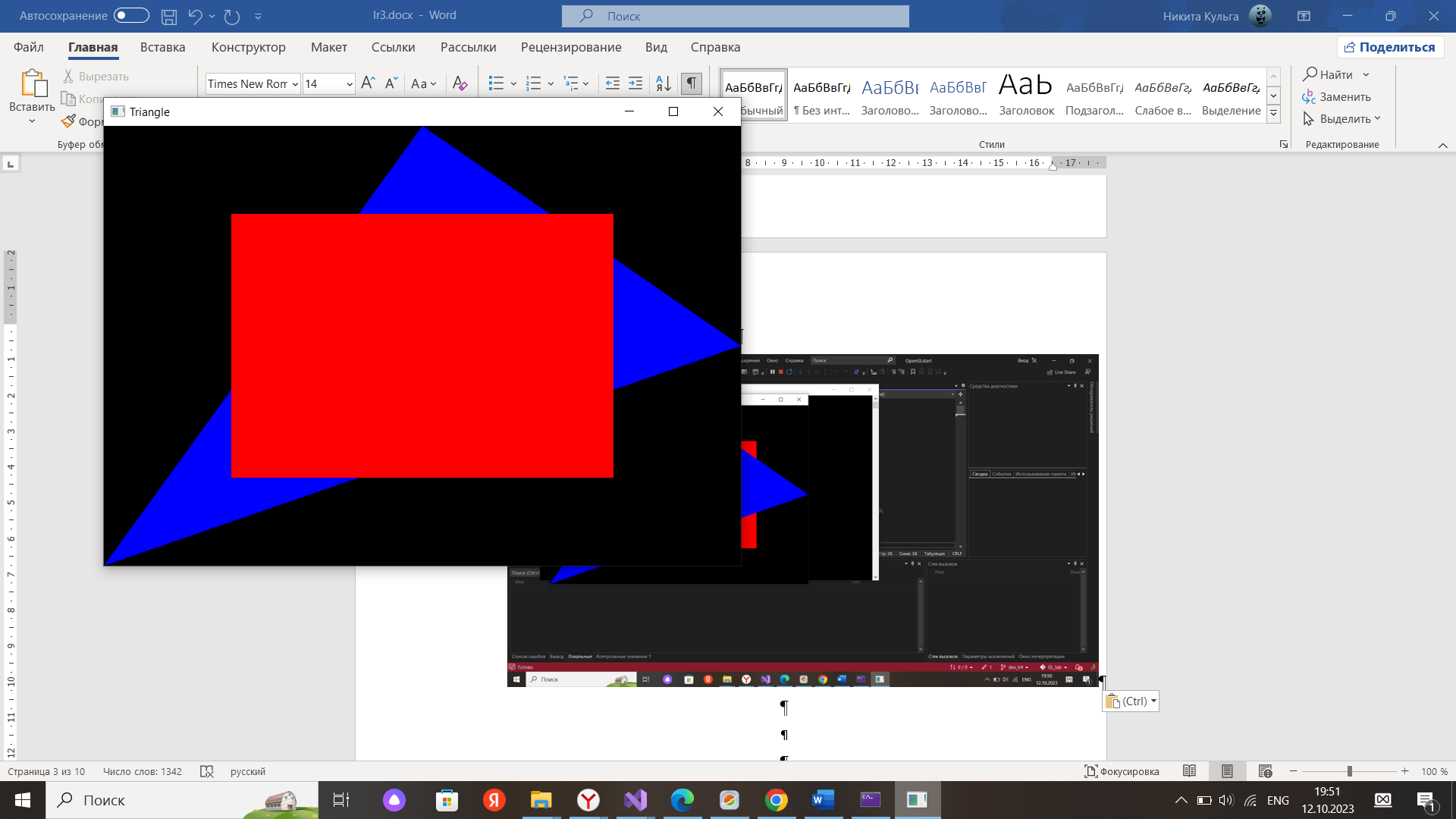
Задание 6.

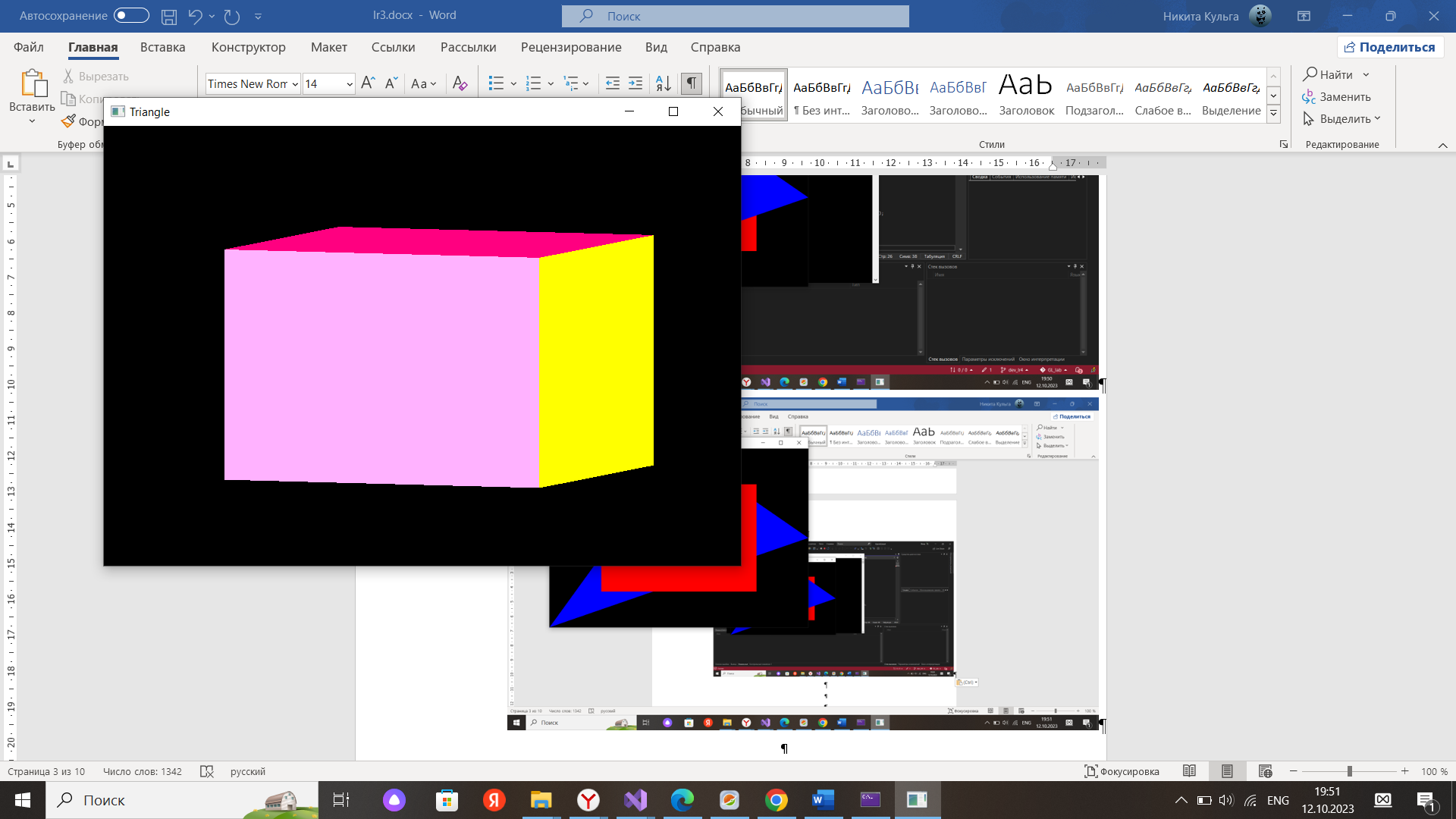
Напишите программу, в которой изображение куба изменяется при нажатии клавиш (в соответствии с вариантом).

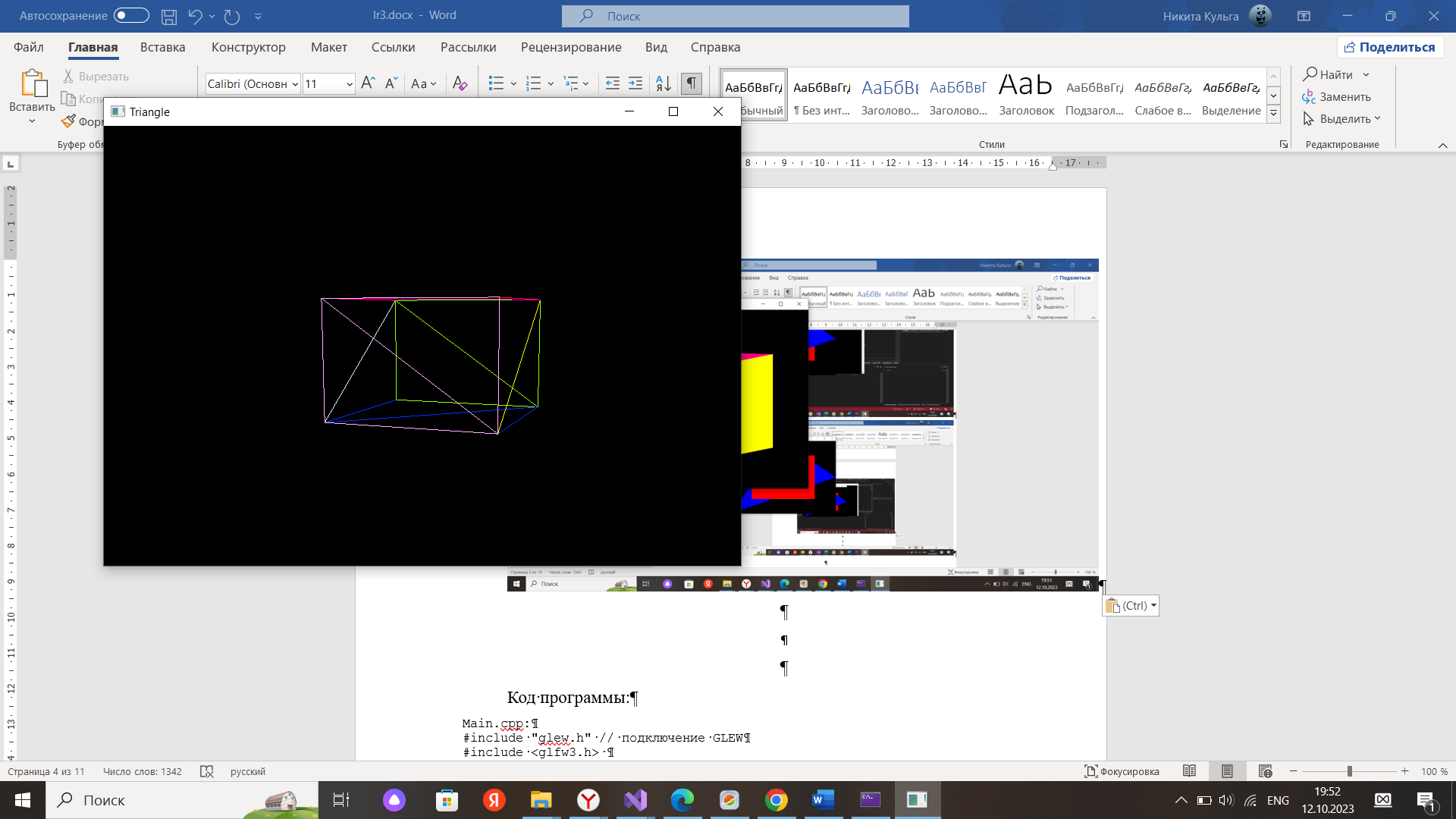


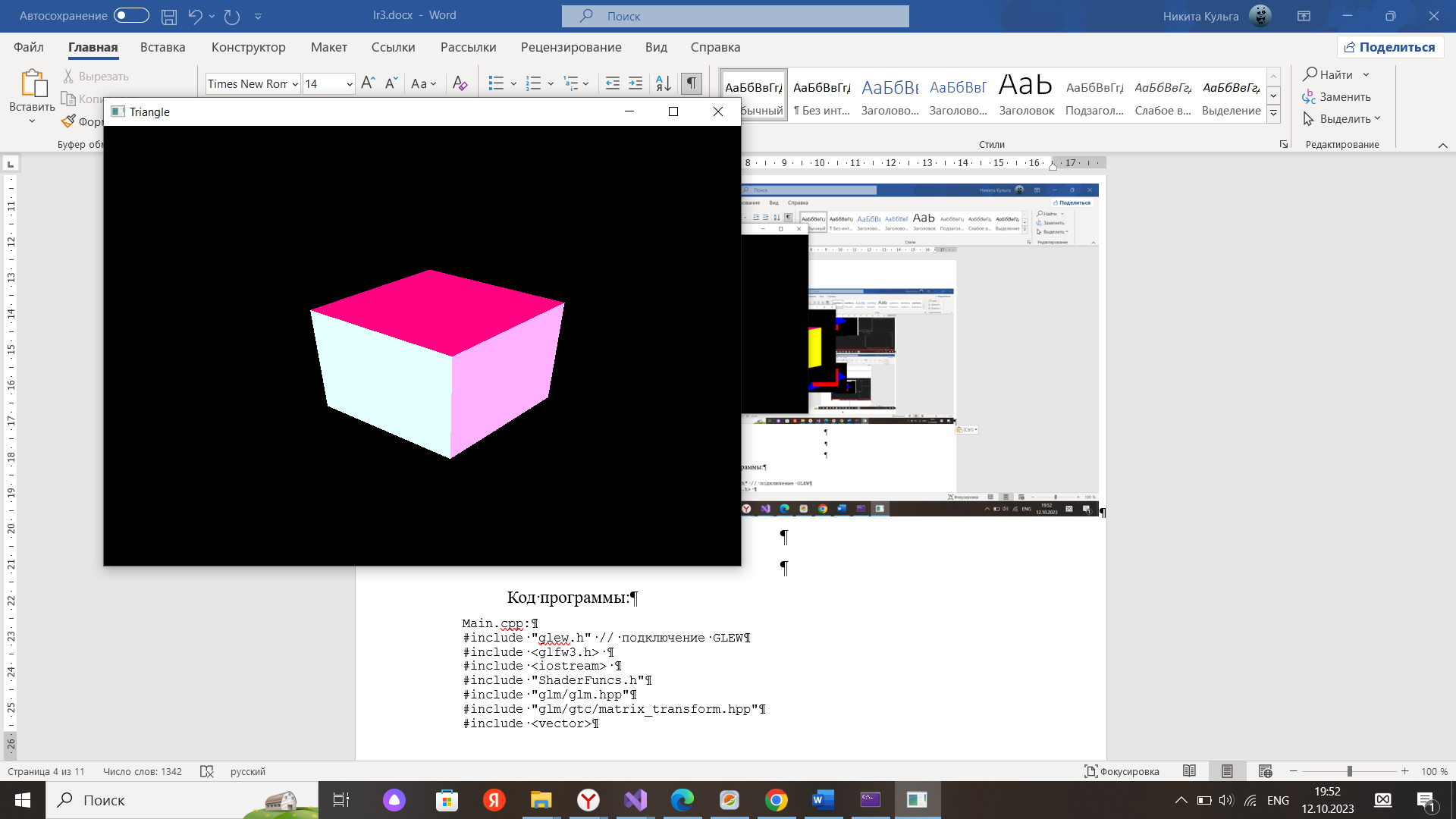
Результат работы программы:

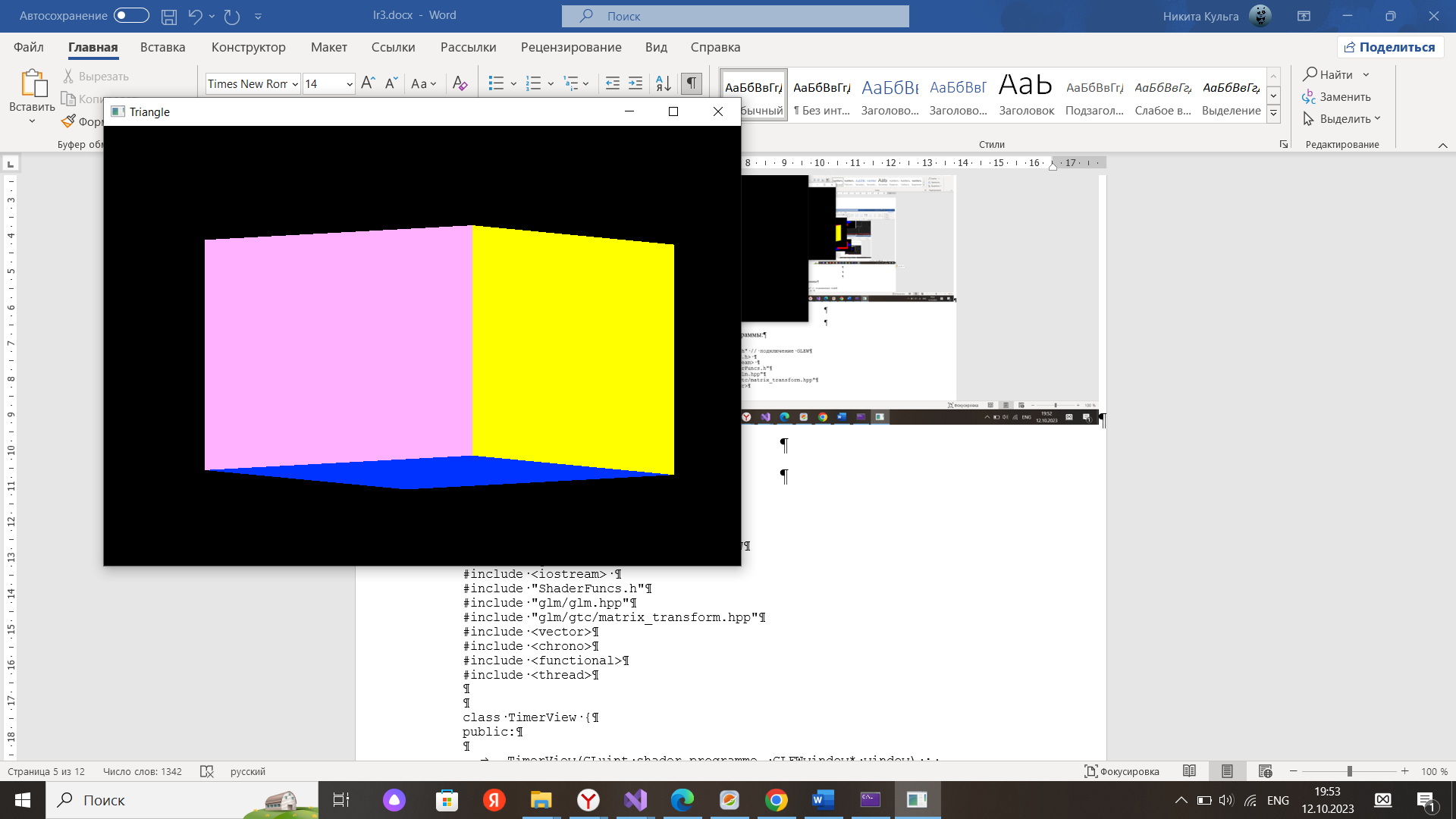


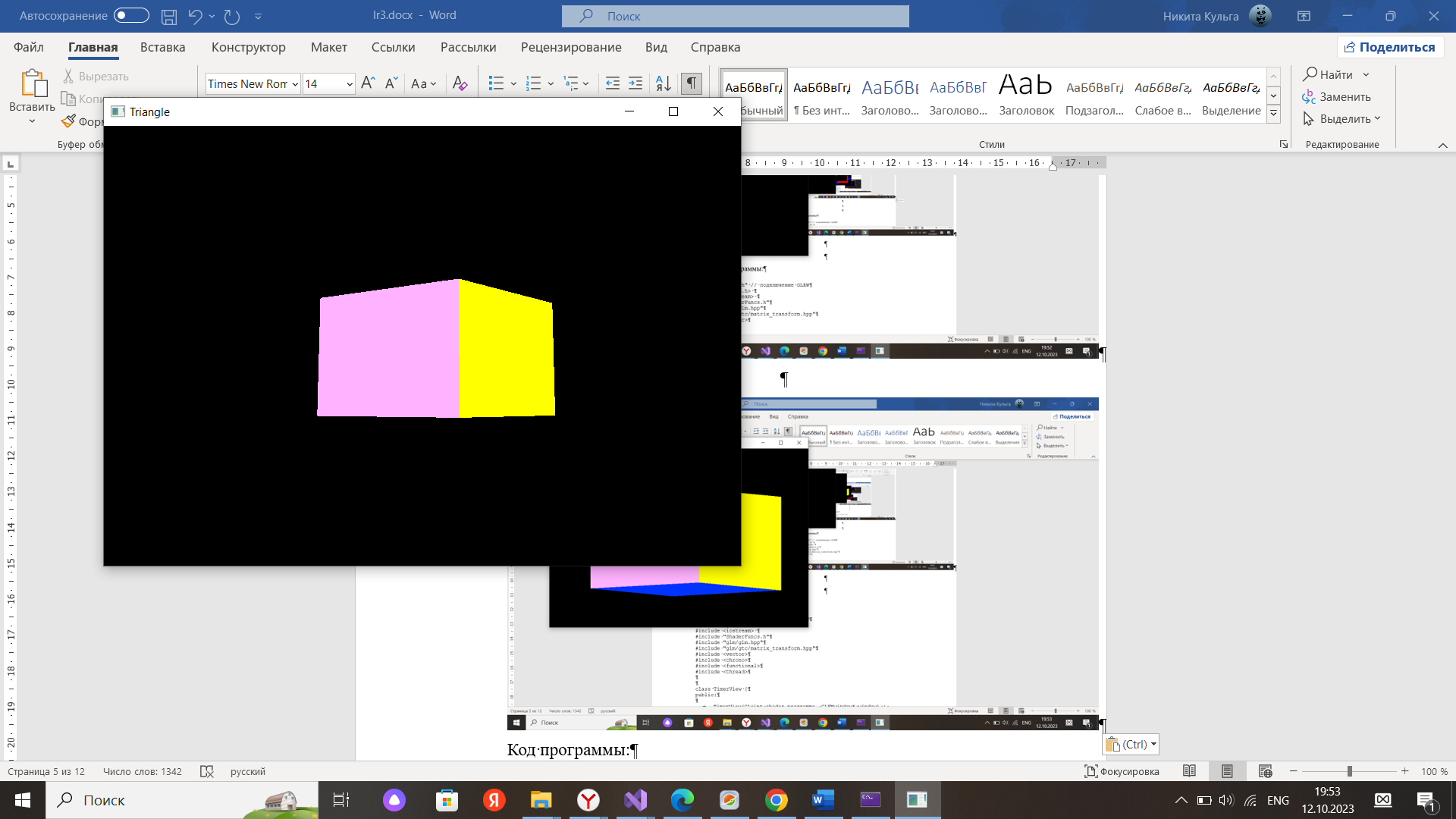












Код программы:

Main.cpp:

#include "glew.h"

#include <glfw3.h>

#include <iostream>

#include "ShaderFuncs.h"

#include "glm/glm.hpp"

#include "glm/gtc/matrix\_transform.hpp"

#include "TimerView.cpp"

#include "dataPoint.h"

int WinWidth = 840;

int WinHeight = 580;

GLfloat x=0;

GLfloat y=0;

bool straightOrPerspective = false;

int numberTask = 0;

void glfw\_window\_size\_callback(GLFWwindow\* window, int width, int height);

void drawSmth(dataPoint dataPoints, std::vector<GLuint> indexes, glm::mat4 transformMatrix, GLuint shader\_programme, GLenum type);

void key\_callback(GLFWwindow\* window, int key, int scancode, int action, int mods);

void task\_1And2(GLint shader\_programme);

void task\_3(GLuint shader\_programme);

void task\_4(GLuint shader\_programme);

void task\_5(GLuint shader\_programme);

void task\_6(GLuint shader\_programme);

void drawCube(GLfloat size, glm::mat4 transformMatrix1, GLuint shader\_programme);

int main() {

glfwInit();

GLFWwindow\* window = glfwCreateWindow(WinWidth, WinHeight, "Triangle", NULL, NULL);

glfwMakeContextCurrent(window);

glewInit();

glfwSetKeyCallback(window, key\_callback);

glfwSetWindowSizeCallback(window, glfw\_window\_size\_callback);

GLuint vs = LoadShader("VertexShader.glsl", GL\_VERTEX\_SHADER);

GLuint fs = LoadShader("FragmentShader.glsl", GL\_FRAGMENT\_SHADER);

GLuint shader\_programme = glCreateProgram();

glAttachShader(shader\_programme, vs);

glAttachShader(shader\_programme, fs);

glBindAttribLocation(shader\_programme, 0, "vertex\_position");

glBindAttribLocation(shader\_programme, 1, "vertex\_colour");

glLinkProgram(shader\_programme);

glViewport(0, 0, WinWidth, WinHeight);

while (!glfwWindowShouldClose(window)) {

glfwPollEvents();

if (GLFW\_PRESS == glfwGetKey(window, GLFW\_KEY\_ESCAPE))

{

glfwSetWindowShouldClose(window, 1);

}

glViewport(0, 0, WinWidth, WinHeight);

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

glUseProgram(shader\_programme);

switch (numberTask) {

case 0:

glDepthFunc(GL\_LESS);

task\_1And2(shader\_programme);

break;

case 1:

glEnable(GL\_DEPTH\_TEST);

glDepthFunc(GL\_LESS);

task\_1And2(shader\_programme);

break;

case 2:

glEnable(GL\_DEPTH\_TEST);

glDepthFunc(GL\_LESS);

task\_3(shader\_programme);

break;

case 3:

glEnable(GL\_DEPTH\_TEST);

glDepthFunc(GL\_LESS);

task\_4(shader\_programme);

break;

case 4:

glEnable(GL\_DEPTH\_TEST);

glDepthFunc(GL\_LESS);

task\_5(shader\_programme);

break;

case 5:

glEnable(GL\_DEPTH\_TEST);

glDepthFunc(GL\_LESS);

task\_6(shader\_programme);

break;

}

glfwSwapBuffers(window);

}

glfwTerminate();

return 0;

}

void key\_callback(GLFWwindow\* window, int key, int scancode, int action, int mods)

{

if (key == GLFW\_KEY\_ESCAPE && action == GLFW\_PRESS)

glfwSetWindowShouldClose(window, 1);

else if (key == GLFW\_KEY\_ENTER && action == GLFW\_PRESS)

numberTask = (numberTask++) % 6;

else if (key == GLFW\_KEY\_DOWN && action == GLFW\_REPEAT)

x -= 1;

else if (key == GLFW\_KEY\_UP && action == GLFW\_REPEAT)

x += 1;

else if (key == GLFW\_KEY\_LEFT && action == GLFW\_REPEAT)

y -= 1;

else if (key == GLFW\_KEY\_RIGHT && action == GLFW\_REPEAT)

y += 1;

else if (key == GLFW\_KEY\_M && action == GLFW\_PRESS)

straightOrPerspective = !straightOrPerspective;

}

void glfw\_window\_size\_callback(GLFWwindow\* window, int width, int

height) {

WinWidth = width;

WinHeight = height;

}

void drawSmth(dataPoint dataPoints, std::vector<GLuint> indexes, glm::mat4 transformMatrix, GLuint shader\_programme, GLenum type) {

GLuint stm = glGetUniformLocation(shader\_programme, "transform");

glUniformMatrix4fv(stm, 1, GL\_FALSE, &transformMatrix[0][0]);

//БУФЕРЫ

GLuint coords\_vbo = 0;

glGenBuffers(1, &coords\_vbo);

glBindBuffer(GL\_ARRAY\_BUFFER, coords\_vbo);

glBufferData(GL\_ARRAY\_BUFFER, dataPoints.sizeDataPoints \* sizeof(GLfloat), dataPoints.getCoordinatesData(), GL\_STATIC\_DRAW);

GLuint colors\_vbo = 0;

glGenBuffers(1, &colors\_vbo);

glBindBuffer(GL\_ARRAY\_BUFFER, colors\_vbo);

glBufferData(GL\_ARRAY\_BUFFER, dataPoints.sizeDataPoints \* sizeof(GLfloat), dataPoints.getColorsData(), GL\_STATIC\_DRAW);

GLuint elementbuffer;

glGenBuffers(1, &elementbuffer);

glBindBuffer(GL\_ELEMENT\_ARRAY\_BUFFER, elementbuffer);

glBufferData(GL\_ELEMENT\_ARRAY\_BUFFER, indexes.size() \* sizeof(GLuint), indexes.data(), GL\_STATIC\_DRAW);

// end БУФЕРЫ

//Массив

GLuint vao = 0;

glGenVertexArrays(1, &vao);

glBindVertexArray(vao);

glBindBuffer(GL\_ARRAY\_BUFFER, coords\_vbo);

glVertexAttribPointer(0, 3, GL\_FLOAT, GL\_FALSE, 0, NULL);

glBindBuffer(GL\_ARRAY\_BUFFER, colors\_vbo);

glVertexAttribPointer(1, 3, GL\_FLOAT, GL\_FALSE, 0, NULL);

glEnableVertexAttribArray(0);

glEnableVertexAttribArray(1);

glBindBuffer(GL\_ELEMENT\_ARRAY\_BUFFER, elementbuffer);

// end массивы

//Отрисовка

glDrawElements(type, indexes.size(), GL\_UNSIGNED\_INT, 0);

// end Отрисовка

//Освобождение памяти

glDeleteVertexArrays(1, &vao);

glDeleteBuffers(1, &coords\_vbo);

glDeleteBuffers(1, &colors\_vbo);

glDeleteBuffers(1, &elementbuffer);

// end Освобождение памяти

}

void task\_1And2(GLint shader\_programme) {

std::vector<GLuint> indexes1 = {

0,1,3,

2,3,1

};

dataPoint dataPoints1;

dataPoints1.addCoordinates({ -0.6f, -0.6f, -0.2f });

dataPoints1.addCoordinates({ 0.6f, -0.6f, -0.2f });

dataPoints1.addCoordinates({ 0.6f, 0.6f, -0.2f });

dataPoints1.addCoordinates({ -0.6f, 0.6f, -0.2f });

dataPoints1.addColors({ 1.0f, 0.0f, 0.0f });

dataPoints1.addColors({ 1.0f, 0.0f, 0.0f });

dataPoints1.addColors({ 1.0f, 0.0f, 0.0f });

dataPoints1.addColors({ 1.0f, 0.0f, 0.0f });

glm::mat4 transformMatrix1 = glm::mat4(1.0f);

drawSmth(dataPoints1, indexes1,transformMatrix1, shader\_programme, GL\_TRIANGLES);

dataPoint dataPoints2;

dataPoints2.addCoordinates({ -0.5f, 0.8f, -0.8f });

dataPoints2.addCoordinates({ -0.5f, 0.8f, 0.8f });

dataPoints2.addCoordinates({ -0.5f, -0.8f, 0.8f });

dataPoints2.addCoordinates({ -0.5f, -0.8f, -0.8f });

dataPoints2.addColors({ 0.0f,1.0f,0.0f });

dataPoints2.addColors({ 0.0f,1.0f,0.0f });

dataPoints2.addColors({ 0.0f,1.0f,0.0f });

dataPoints2.addColors({ 0.0f,1.0f,0.0f });

std::vector<GLuint> indexes2 = {

0,2,1,

0,2,3

};

glm::mat4 transformMatrix2 = glm::mat4(1.0f);

drawSmth(dataPoints2, indexes2, transformMatrix2, shader\_programme, GL\_TRIANGLES);

dataPoint dataPoints3;

dataPoints3.addCoordinates({ -1.0f, -1.0f, 0.0f });

dataPoints3.addCoordinates({ 0.0f, 1.0f, 1.0f });

dataPoints3.addCoordinates({ 1.0f, 0.0f, 0.0f });

dataPoints3.addColors({ 0.0f,0.0f,1.0f });

dataPoints3.addColors({ 0.0f,0.0f,1.0f });

dataPoints3.addColors({ 0.0f,0.0f,1.0f });

std::vector<GLuint> indexes3 = {0,2,1,};

glm::mat4 transformMatrix3 = glm::mat4(1.0f);

drawSmth(dataPoints3, indexes3, transformMatrix3, shader\_programme, GL\_TRIANGLES);

}

void task\_3(GLuint shader\_programme) {

glm::mat4 transformMatrix1 = glm::mat4(1.0f);

transformMatrix1 = glm::translate(transformMatrix1, { 0.0f,0.0f,-4.0f });

glm::mat4 projMatrix = glm::ortho(-1.0f, 0.9f, -0.9f, 1.0f, 2.0f, 9.0f);

transformMatrix1 = glm::rotate(transformMatrix1, glm::radians(x), glm::vec3(1.0, 0.0, 0.0));

transformMatrix1 = glm::rotate(transformMatrix1, glm::radians(y), glm::vec3(0.0, 1.0, 0.0));

drawCube(0.5f, projMatrix \* transformMatrix1, shader\_programme);

}

void task\_4(GLuint shader\_programme) {

glm::mat4 transformMatrix1 = glm::mat4(1.0f);

transformMatrix1 = glm::translate(transformMatrix1, { 0.0f,0.0f,-4.0f });

glm::mat4 projMatrix = glm::frustum(-1.0f, 0.9f, -0.9f, 1.0f, 2.0f, 9.0f);

transformMatrix1 = glm::rotate(transformMatrix1, glm::radians(x), glm::vec3(1.0, 0.0, 0.0));

transformMatrix1 = glm::rotate(transformMatrix1, glm::radians(y), glm::vec3(0.0, 1.0, 0.0));

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_LINE);

drawCube(0.5f, projMatrix \* transformMatrix1, shader\_programme);

glPolygonMode(GL\_FRONT\_AND\_BACK, GL\_FILL);

}

void task\_5(GLuint shader\_programme) {

glm::mat4 transformMatrix1 = glm::mat4(1.0f);

transformMatrix1 = glm::translate(transformMatrix1, { 0.0f,0.0f,-4.0f });

glm::mat4 projMatrix = glm::frustum(-1.0f, 0.9f, -0.9f, 1.0f, 2.0f, 9.0f);

transformMatrix1 = glm::rotate(transformMatrix1, glm::radians(35.0f), glm::vec3(1.0, 0.0, 0.0));

transformMatrix1 = glm::rotate(transformMatrix1, glm::radians(50.0f), glm::vec3(0.0, 1.0, 0.0));

drawCube(0.5f, projMatrix \* transformMatrix1, shader\_programme);

}

void task\_6(GLuint shader\_programme) {

glm::mat4 transformMatrix1 = glm::mat4(1.0f);

transformMatrix1 = glm::translate(transformMatrix1, { 0.0f,0.0f,-4.0f });

transformMatrix1 = glm::rotate(transformMatrix1, glm::radians(x), glm::vec3(1.0, 0.0, 0.0));

transformMatrix1 = glm::rotate(transformMatrix1, glm::radians(y), glm::vec3(0.0, 1.0, 0.0));

glm::mat4 projMatrix = straightOrPerspective==true ? glm::frustum(-1.0f, 0.9f, -0.9f, 1.0f, 2.0f, 9.0f) : glm::ortho(-1.0f, 0.9f, -0.9f, 1.0f, 2.0f, 9.0f);

drawCube(0.5f, projMatrix \* transformMatrix1, shader\_programme);

}

void drawCube(GLfloat position, glm::mat4 transformMatrix, GLuint shader\_programme) {

dataPoint dataPoints;

dataPoints.addCoordinates({ position,position,position });

dataPoints.addCoordinates({ -position,position,position });

dataPoints.addCoordinates({ position,-position,position });

dataPoints.addCoordinates({ -position,-position,position });

dataPoints.addCoordinates({ position,position,-position });

dataPoints.addCoordinates({ -position,position,-position });

dataPoints.addCoordinates({ position,-position,-position });

dataPoints.addCoordinates({ -position,-position,-position });

dataPoints.addCoordinates({ position,position,position });

dataPoints.addCoordinates({-position,position,position});

dataPoints.addCoordinates({position,position,-position});

dataPoints.addCoordinates({ -position,position,-position});

dataPoints.addCoordinates({ position,-position,position });

dataPoints.addCoordinates({ -position,-position,position });

dataPoints.addCoordinates({ position,-position,-position });

dataPoints.addCoordinates({ -position,-position,-position });

dataPoints.addCoordinates({ position,position,position });

dataPoints.addCoordinates({ position,-position,position });

dataPoints.addCoordinates({ position,position,-position });

dataPoints.addCoordinates({ position,-position,-position });

dataPoints.addCoordinates({ -position,position,position });

dataPoints.addCoordinates({ -position,-position,position });

dataPoints.addCoordinates({ -position,position,-position });

dataPoints.addCoordinates({ -position,-position,-position });

dataPoints.addColors({ 1.0f,0.7f,1.0f});

dataPoints.addColors({1.0f,0.7f,1.0f});

dataPoints.addColors({1.0f,0.7f,1.0f});

dataPoints.addColors({1.0f,0.7f,1.0f});

dataPoints.addColors({0.6f,1.0f,0.0f});

dataPoints.addColors({0.6f,1.0f,0.0f});

dataPoints.addColors({0.6f,1.0f,0.0f});

dataPoints.addColors({0.6f,1.0f,0.0f});

dataPoints.addColors({1.0f,0.0f,0.5f});

dataPoints.addColors({1.0f,0.0f,0.5f});

dataPoints.addColors({1.0f,0.0f,0.5f});

dataPoints.addColors({1.0f,0.0f,0.5f});

dataPoints.addColors({0.0f,0.2f,1.0f});

dataPoints.addColors({0.0f,0.2f,1.0f});

dataPoints.addColors({0.0f,0.2f,1.0f});

dataPoints.addColors({0.0f,0.2f,1.0f});

dataPoints.addColors({1.0f,1.0f,0.0f});

dataPoints.addColors({1.0f,1.0f,0.0f});

dataPoints.addColors({1.0f,1.0f,0.0f});

dataPoints.addColors({1.0f,1.0f,0.0f});

dataPoints.addColors({0.9f,1.0f,1.0f});

dataPoints.addColors({0.9f,1.0f,1.0f});

dataPoints.addColors({0.9f,1.0f,1.0f});

dataPoints.addColors({0.9f,1.0f,1.0f});

std::vector<GLuint> indexes = { 0,1,2,

3,1,2,

4,5,6,

7,5,6,

8,9,10,

11,9,10,

12,13,14,

15,13,14,

16,17,18,

19,17,18,

20,21,22,

23,21,22,

};

drawSmth(dataPoints, indexes, transformMatrix, shader\_programme, GL\_TRIANGLES);

}

datapoint.h

#pragma once

#include <vector>

#include "glew.h" // подключение GLEW

#include <glfw3.h>

#include "glm/glm.hpp"

#include "glm/gtc/matrix\_transform.hpp"

struct dataPoint {

std::vector<glm::vec3> coordinates;

std::vector<glm::vec3> colors;

size\_t sizeDataPoints;

void addCoordinates(glm::vec3 coordinate) {

coordinates.push\_back(coordinate);

sizeDataPoints = (coordinates.size()) == 0 ? 0 : 3 \* coordinates.size();

}

void addColors(glm::vec3 color) {

colors.push\_back(color);

}

// Метод для получения указателя на данные координат

glm::vec3\* getCoordinatesData() {

return coordinates.data();

}

// Метод для получения указателя на данные цветов

glm::vec3\* getColorsData() {

return colors.data();

}

};

Shaderfuncs.h:

#pragma once

#include <iostream>

#include <string>

#include <GL/glew.h>

#include <GLFW/glfw3.h>

GLuint LoadShader(std::string filename, GLenum type);

Shaderfuncs.cpp:

#include "ShaderFuncs.h"

#include <fstream>

#include <vector>

GLuint LoadShader(std::string filename, GLenum type)

{

std::ifstream fin;

fin.open(filename);

if (fin) {

std::string textshader;

char c= fin.get();

while (c!=-1) {

textshader += c;

c = fin.get();

}

fin.close();

GLuint s = glCreateShader(type);

char\* buf = new GLchar[textshader.length() + 1];

strcpy\_s(buf, textshader.length()+1 , textshader.c\_str());

glShaderSource(s, 1, &buf, NULL);

glCompileShader(s);

GLint isCompiled = 0;

glGetShaderiv(s, GL\_COMPILE\_STATUS, &isCompiled);

if (isCompiled == GL\_FALSE)

{

GLint maxLength = 0;

glGetShaderiv(s, GL\_INFO\_LOG\_LENGTH, &maxLength);

std::vector<GLchar> errorLog(maxLength);

glGetShaderInfoLog(s, maxLength, &maxLength, &errorLog[0]);

for (GLchar c : errorLog)

std::cout << c;

}

delete[] buf;

return s;

}

else {

std::cout << "ERROR: No such file";

return -1;

}

}

FragmentShader.glsl:

#version 400

in vec3 colour;

out vec4 frag\_colour;

void main () {

frag\_colour = vec4 (colour, 1.0);

}

VertexShaner.glsl:

#version 400

in vec3 vertex\_position;

in vec3 vertex\_colour;

uniform mat4 transform;

out vec3 colour;

void main () {

colour = vertex\_colour;

gl\_Position = transform \* vec4 (vertex\_position, 1.0);

}