Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика, искусственный интеллект и системы управления»

КАФЕДРА «Прикладная математика и информатика»

Лабораторная работа № 2

по курсу «Алгоритмы компьютерной графики»

Студент группы ИУ9-41Б Горбунов А. Д.

Преподаватель Цапкович П. А.

Москва 2024

1 Цель

Целью работы является знакомство с библиотекой OpenGL, принципами разработки алгоритмов копмьютерной графики и их реализацией на языке C++.

2 правильная призма

1.Определить куб в качестве модели объекта сцены.

2.Определить преобразования, позволяющие получить заданный вид проекции (в соответствии с вариантом). Для демонстрации проекции добавить в сцену куб (в стандартной ориентации, не изменяемой при модельно-видовых преобразованиях основного объекта).

3.Реализовать изменение ориентации и размеров объекта (навигацию камеры) с помощью модельно-видовых преобразований (без gluLookAt). Управление производится интерактивно с помощью клавиатуры и/или мыши.

4.Предусмотреть возможность переключения между каркасным и твердотельным отображением модели (glFrontFace/ glPolygonMode).

3 Практическая реализация

```
#include <GLFW/glfw3.h>
#include <cmath>

using std::cos, std::sin;

int mode = 0;
float x = 0.0f;
float y = 0.0f;

void key_callback(GLFWwindow* window, int key, int scancode, int action, int mods
    if (action == GLFW_PRESS || action == GLFW_REPEAT)
    {
        if (key == GLFW_KEY_RIGHT) { y += 0.25f; }
        if (key == GLFW_KEY_LEFT) {  y -= 0.25f; }
        if (key == GLFW_KEY_UP) {    x += 0.25f; }
        if (key == GLFW_KEY_DOWN) {  x -= 0.25f; }
        if (key == GLFW_KEY_SPACE) { mode = (mode + 1) % 2; }
    }
}
void cube() {
    glBegin(GL_QUADS);
    glColor3f(1.0f, 0.3f, 0.3f);
    glVertex3f(0.5f, -0.5f, -0.5f);
    glVertex3f(0.5f, 0.5f, -0.5f);
    glVertex3f(-0.5f, 0.5f, -0.5f);
    glVertex3f(-0.5f, -0.5f, -0.5f);
    glEnd();

    glBegin(GL_QUADS);
    glColor3f(0.5f, 0.7f, 0.7f);
    glVertex3f(0.5f, -0.5f, 0.5f);
```

```
glVertex3f(0.5f, 0.5f, 0.5f);
glVertex3f(-0.5f, 0.5f, 0.5f);
glVertex3f(-0.5f, -0.5f, 0.5f);
glEnd();

glBegin(GL_QUADS);
glColor3f(0.4f, 0.4f, 1.0f);
glVertex3f(0.5f, -0.5f, -0.5f);
glVertex3f(0.5f, 0.5f, -0.5f);
glVertex3f(0.5f, 0.5f, 0.5f);
glVertex3f(0.5f, -0.5f, 0.5f);
glEnd();

glBegin(GL_QUADS);
glColor3f(0.5f, 1.0f, 0.5f);
glVertex3f(-0.5f, -0.5f, 0.5f);
glVertex3f(-0.5f, 0.5f, 0.5f);
glVertex3f(-0.5f, 0.5f, -0.5f);
glVertex3f(-0.5f, -0.5f, -0.5f);
glEnd();

glBegin(GL_QUADS);
glColor3f(0.1f, 0.5f, 0.5f);
glVertex3f(0.5f, 0.5f, 0.5f);
glVertex3f(0.5f, 0.5f, -0.5f);
glVertex3f(-0.5f, 0.5f, -0.5f);
glVertex3f(-0.5f, 0.5f, 0.5f);
glEnd();

glBegin(GL_QUADS);
glColor3f(1.0f, 0.4f, 0.5f);
glVertex3f(0.5f, -0.5f, -0.5f);
glVertex3f(0.5f, -0.5f, 0.5f);
```

```
    glVertex3f(-0.5f, -0.5f, 0.5f);
    glVertex3f(-0.5f, -0.5f, -0.5f);
    glEnd();
}

void display(GLFWwindow* window) {
    glEnable(GL_DEPTH_TEST);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glViewport(640 - 160, 640 - 160, 160, 160);
    glLoadIdentity();
    glClearColor(0, 0, 0, 0);

    float rotate_x[] = {
        1, 0, 0, 0,
        0, cos(x), sin(x), 0,
        0, -sin(x), cos(x), 0,
        0, 0, 0, 1
    };

    float rotate_y[] = {
        cos(y), 0, -sin(y), 0,
        0, 1, 0, 0,
        sin(y), 0, cos(y), 0,
        0, 0, 0, 1
    };

    float mat1[] = {
        1, 0, 0, 0,
        0, 1, 0, 0,
        0, 0, -1, 0,
        0, 0, 0, 1
    };
```

```
float mat2[] = {
    0, 0, -1, 0,
    0, 1, 0, 0,
    -1, 0, 0, 0,
    0, 0, 0, 1
};

float mat3[] = {
    1, 0, 0, 0,
    0, 0, -1, 0,
    0, -1, 0, 0,
    0, 0, 0, 1
};

if (mode == 0) {
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
} else {
    glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
}

glPushMatrix();

cube();
glRotatef(20, 20, 20, 45);
glPopMatrix();

glViewport(640 - 3 * 160, 640 - 160, 160, 160);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glMultMatrixf(mat1);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
```

```
    glMultMatrixf(rotate_x);
    glMultMatrixf(rotate_y);
    cube();

    glViewport(640 - 3 * 160, 640 - 3 * 160, 160, 160);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glMultMatrixf(mat2);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glMultMatrixf(rotate_x);
    glMultMatrixf(rotate_y);
    cube();

    glViewport(640 - 160, 640 - 3 * 160, 160, 160);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glMultMatrixf(mat3);

    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glMultMatrixf(rotate_x);
    glMultMatrixf(rotate_y);
    cube();

    glfwSwapBuffers(window);
    glfwPollEvents();
}

int main() {
    if (!glfwInit()) {
        return -1;
```

```
    }

    GLFWwindow* window = glfwCreateWindow(800, 800, "Lab2", NULL, NULL);
    if (!window) {
        glfwTerminate();
        return -1;
    }

    glfwMakeContextCurrent(window);
    glfwSetKeyCallback(window, key_callback);

    while (!glfwWindowShouldClose(window)) {
        display(window);
    }

    glfwDestroyWindow(window);
    glfwTerminate();
    return 0;
}
```

4 Вывод

В данной работе я изучил возможности языка c++ и библиотеки OpenGL, приобрёл навыки разработки на языке c++ алгоритмов копьютерной графики, углубил свои знания в программировании и изучил подробнее устройство копьютерной графики.

5 Результат запуска

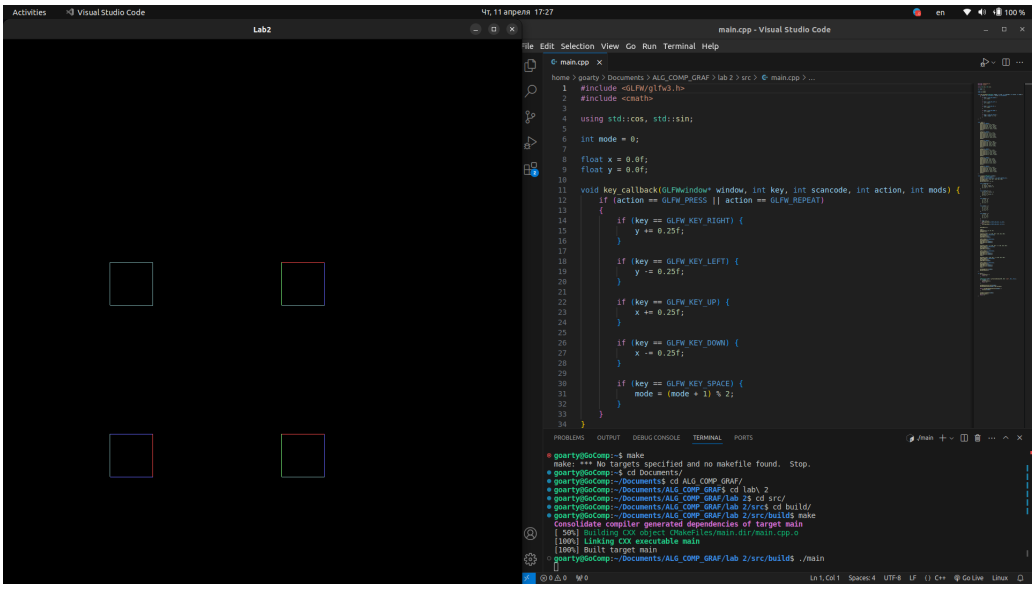

Рис. 1 — Без воздействия

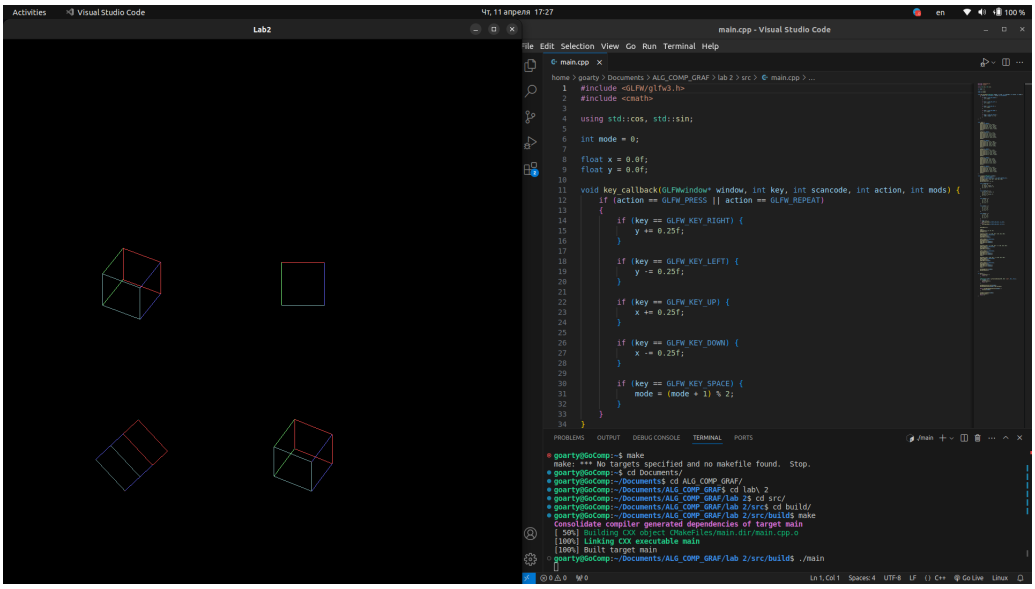

Рис. 2 — Повёрнутый

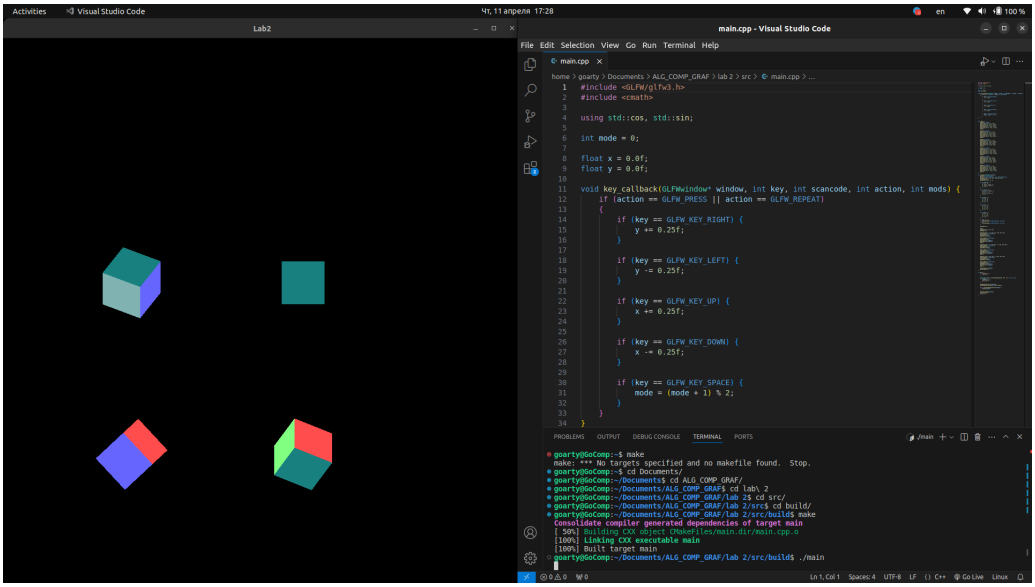

Рис. 3 — заполненый