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 и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 2

по курсу «Компьютерная Графика»

Студент группы ИУ9-42Б Павлов И. П.

Преподаватель Цалкович П.А.

Москва 2024

1 Задача

- Определить куб в качестве модели объекта сцены
- 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).
- Реализовать изменение ориентации и размеров объекта (навигацию камеры) с помощью модельно-видовых преобразований (без gluLookAt). Управление производится интерактивно с помощью клавиатуры и/или мыши.
- Предусмотреть возможность переключения между каркасным и твердотельным отображением модели (glFrontFace/glPolygonMode).

В соответствии с вариантом было необходимо реализовать трёхточечную перспективу.

2 Основная теория

Для выполнения задания было необходимо добавить куб в качестве объекта сцены без поддержки модельно-видовых образований, а также добавить второй куб, для преобразований которого были использованы следующие матрицы:

- Матрица трансляции

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -0.5 & -0.5 & 0 & 1 \end{pmatrix}$$

- Матрица проекции

$$\begin{pmatrix} 1 & 0 & 0 & 0.8 \\ 0 & 1 & 0 & 0.8 \\ 0 & 0 & 1 & 0.8 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Матрица поворота относительно Ox

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Матрица поворота относительно Oy

$$\begin{pmatrix} \cos(\beta) & 0 & \sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Помимо реализации преобразований была добавлена возможность включать и отключать заливку.

3 Практическая реализация

3.1 Реализация main.cpp

```
#include "main.h"
#include <cmath>

int width = 750;
int height = 750;
float alpha = 0.7f;
float beta = 0.8f;
bool fill = true;

using std::sin, std::cos;

int main() {
    if (!glfwInit()) {
        return 1;
    }

    GLFWwindow* window = glfwCreateWindow(width, height, "Lab2",
                                          nullptr, nullptr);

    if (window == nullptr) {
        glfwTerminate();
        return 1;
    }

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

```
    glEnable(GL_DEPTH_TEST);
    glDepthFunc(GL_LESS);

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

    glfwDestroyWindow(window);
    glfwTerminate();

    return 0;
}

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) {
            beta += 0.1;
        } else if (key == GLFW_KEY_LEFT) {
            beta -= 0.1;
        } else if (key == GLFW_KEY_UP) {
            alpha += 0.1;
        } else if (key == GLFW_KEY_DOWN) {
            alpha -= 0.1;
        } else if (key == GLFW_KEY_F) {
            fill = !fill;
            if (fill) {
                glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
            } else {
                glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
            }
        }
    }
}

void draw_cube(float p) {
    glBegin(GL_QUADS);
    glColor3f(0.0, 0.0, 1.0);
    glVertex3f(-p/2, -p/2, -p/2);
    glVertex3f(-p/2,  p/2, -p/2);
    glVertex3f(-p/2,  p/2,  p/2);
    glVertex3f(-p/2, -p/2,  p/2);
    glColor3f(1.0, 0.0, 0.0);
    glVertex3f( p/2, -p/2, -p/2);
    glVertex3f( p/2, -p/2,  p/2);
    glVertex3f( p/2,  p/2,  p/2);
    glVertex3f( p/2,  p/2, -p/2);
    glColor3f(0.0, 1.0, 0.0);
    glVertex3f(-p/2, -p/2, -p/2);
    glVertex3f(-p/2, -p/2,  p/2);
    glVertex3f( p/2, -p/2,  p/2);
    glVertex3f( p/2, -p/2, -p/2);
    glColor3f(1.0, 1.0, 0.0);
    glVertex3f(-p/2, p/2, -p/2);
    glVertex3f(-p/2, p/2,  p/2);
    glVertex3f( p/2, p/2,  p/2);
    glVertex3f( p/2, p/2, -p/2);
    glColor3f(0.0, 1.0, 1.0);
    glVertex3f(-p/2, -p/2, -p/2);
```

```
    glVertex3f( p/2, -p/2, -p/2);
    glVertex3f( p/2,  p/2, -p/2);
    glVertex3f(-p/2,  p/2, -p/2);
    glColor3f(1.0, 0.0, 1.0);
    glVertex3f(-p/2, -p/2,  p/2);
    glVertex3f( p/2, -p/2,  p/2);
    glVertex3f( p/2,  p/2,  p/2);
    glVertex3f(-p/2,  p/2,  p/2);
    glEnd();
}

GLfloat proj[] = {
        1, 0, 0, 0.8,
        0, 1, 0, 0.8,
        0, 0, 1, 0.8,
        0, 0, 0, 1
};

GLfloat trans1[] = {
        1, 0, 0, 0,
        0, 1, 0,0,
        0, 0, 1, 0,
        -0.5, -0.5, 0, 1
};

GLfloat trans2[] = {
        1, 0, 0, 0,
        0, 1, 0,0,
        0, 0, 1, 0,
        0.5, 0.5, 0, 1
};

void display(GLFWwindow* window) {
    glClear(GL_COLOR_BUFFER_BIT);
    glClear(GL_DEPTH_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);

    glLoadIdentity();
    glPushMatrix();
    glMatrixMode(GL_PROJECTION);

    glMultMatrixf(trans1);
    glMultMatrixf(proj);
    GLfloat rotX1[] = {
            1, 0, 0, 0,
            0, cos(alpha), -sin(alpha), 0,
            0, sin(alpha), cos(alpha), 0,
            0, 0, 0, 1
    };
    GLfloat rotY1[] = {
            cos(beta), 0, sin(beta), 0,
            0, 1, 0, 0,
            -sin(beta), 0, cos(beta), 0,
            0, 0, 0, 1
    };
    glMultMatrixf(rotX1);
    glMultMatrixf(rotY1);
    draw_cube(0.3);
    glPopMatrix();
```

```
    glPushMatrix();
    glMatrixMode(GL_PROJECTION);
    glMultMatrixf(trans2);
    GLfloat rotX2[] = {
            1, 0, 0, 0,
            0, cos(0.7f), -sin(0.7f), 0,
            0, sin(0.7f), cos(0.7f), 0,
            0, 0, 0, 1
    };
    GLfloat rotY2[] = {
            cos(0.7f), 0, sin(0.7f), 0,
            0, 1, 0, 0,
            -sin(0.7f), 0, cos(0.7f), 0,
            0, 0, 0, 1
    };
    glMultMatrixf(rotX2);
    glMultMatrixf(rotY2);
    draw_cube(0.3);
    glPopMatrix();

    glfwSwapBuffers(window);
    glfwPollEvents();
}
```

3.2 Реализация main.h

```
#ifndef PROGRAM_MAIN_H
#define PROGRAM_MAIN_H

#include <cstdlib>
#include "GLFW/glfw3.h"

int main();
void key_callback(GLFWwindow*, int, int, int, int);
void draw_cube(float);
void display(GLFWwindow*);

#endif //PROGRAM_MAIN_H
```

4 Заключение

Во время выполнения лабораторной работы получен навык работы с трёхмерными объектами. Были изучены способы применения матриц трансформации, а также выбраны наиболее подходящие матрицы для конкретного случая. Изучено понятие перспективной проекции, была реализована отрисовка этой проекции на примере куба. Реализована возможность отображения куба и в обычном, и в скелетном видах. Добавлена поддержка управления преобразованиями с клавиатуры.