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)

Отчёт по лабораторной работе № 2 «Модельно-видовые и проективные преобразования»

по курсу

«Алгоритмы компьютерной графики»

Студент группы ИУ9-43Б Бакланова А.Д.

Преподаватель Вишняков И.Э.

2019 г.

СОДЕРЖАНИЕ

1. Постановка задачи

Цель работы: изучение модельно-видовых и проективных преобразований, а также приобретение навыков формирования матриц преобразований в OpenGL.

1. 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).
2. Реализовать изменение ориентации и размеров объекта (навигацию камеры) с помощью модельно-видовых преобразований. Управление производится интерактивно с помощью клавиатуры и/или мыши.
3. Предусмотреть возможность переключения между каркасным и твердотельным отображением модели (glFrontFace / glPolygonMode).

2. Построение куба и применение к нему проекции

Преобразование, данное в 3 варианте - диметрия.

Одна ортографическая проекция не может дать представления об общей трехмерной форме объекта. Это ограничение можно преодолеть с помощью аксонометрических проекций.

Аксонометрическая проекция образуется манипулированием объекта с помощью поворотов и перемещений таким образом, что бы были видны по крайней мере три соседние грани.

Диметрическая проекция - это аксонометрическая проекция, у которой коэффициенты искажения по двум осям имеют равные значения, а искажение по третьей оси может принимать иное значение. Диметрическая проекция строится с помощью поворота на угол ϕ вокруг оси y, затем поворота на угол θ вокруг оси x и проецирования на плоскость $z = 0$ с центром проекции, расположенным в бесконечности.

Диметрия задается формулами (1) и (2):

$$\theta = \arcsin(\pm f_z/\sqrt{2}) \tag{1}$$

$$\phi = \arcsin(\pm f_z/\sqrt{2 - f_z^2}) \tag{2}$$

Диапазон коэффициентов искажения равен $0 \leq f \leq 1$. На рис. 1 показаны разные диметрические проекции для различных значений коэффициента искажения.

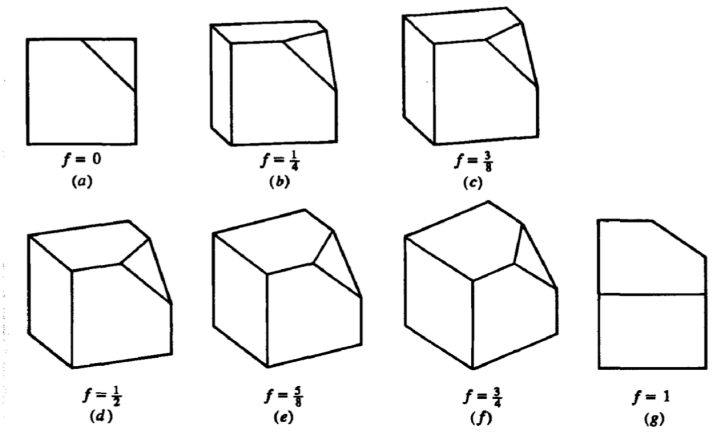

Рисунок 1. Диметрические проекции для различных значений коэффициента искажения.

Прямоугольные аксонометрические проекции строятся по формуле (3):

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3}$$

Лабораторная выполнена при коэффициенте искажения $f = 5/8$.
Функция, которая рисует куб:

```
void drawCube() {
  glBegin(GL_POLYGON);
      glColor3f(1.0,  0.0,  0.0);
    glVertex3f( 0.4, -0.4, -0.4);
    glVertex3f( 0.4,  0.4, -0.4);
      glColor3f(0.6,  0.6,  0.3);
    glVertex3f(-0.4,  0.4, -0.4);
    glVertex3f(-0.4, -0.4, -0.4);
    glEnd();

    glBegin(GL_POLYGON);
      glColor3f(0.01,  1.0, 0.8);
    glVertex3f( 0.4, -0.4, 0.4);
    glVertex3f( 0.4,  0.4, 0.4);
      glColor3f(0.09,  0.56, 0.21);
    glVertex3f(-0.4,  0.4, 0.4);
    glVertex3f(-0.4, -0.4, 0.4);
    glEnd();

    glBegin(GL_POLYGON);
      glColor3f(0.04,  0.14,  0.6);
```

```
glVertex3f(0.4, -0.4, -0.4);
glVertex3f(0.4,  0.4, -0.4);
    glColor3f(1.0,  0.0,  1.0);
glVertex3f(0.4,  0.4,  0.4);
glVertex3f(0.4, -0.4,  0.4);
glEnd();

glBegin(GL_POLYGON);
    glColor3f(0.8,  0.9,  0.04);
glVertex3f(-0.4, -0.4,  0.4);
glVertex3f(-0.4,  0.4,  0.4);
    glColor3f(0.9,  0.4,  0.02);
glVertex3f(-0.4,  0.4, -0.4);
glVertex3f(-0.4, -0.4, -0.4);
glEnd();

glBegin(GL_POLYGON);
    glColor3f(0.3,  0.0,  0.7);
glVertex3f( 0.4,  0.4,  0.4);
glVertex3f( 0.4,  0.4, -0.4);
    glColor3f(0.7,  0.3,  0.9);
glVertex3f(-0.4,  0.4, -0.4);
glVertex3f(-0.4,  0.4,  0.4);
glEnd();

glBegin(GL_POLYGON);
    glColor3f(1.0,  0.9,  0.2);
glVertex3f( 0.4, -0.4, -0.4);
glVertex3f( 0.4, -0.4,  0.4);
    glColor3f(0.8,  0.9,  0.5);
```

```
        glVertex3f(-0.4, -0.4,  0.4);
        glVertex3f(-0.4, -0.4, -0.4);
        glEnd();
    }
```

Данные для проекции:

```
    float fi = (asin(0.625/sqrt(2.0-0.625*0.625)));
    float teta = (asin(0.625/sqrt(2.0)));
    GLfloat m[16] = {
    cos(fi),  sin(fi)*sin(teta),  sin(fi)*cos(teta), 0,
          0,          cos(teta),         -sin(teta), 0,
    sin(fi), -cos(fi)*sin(teta), -cos(fi)*cos(teta), 0,
          0,                0,                0, 1
    };

    int main(int argc, char const *argv[]) {

        if(!glfwInit()) {
            exit(EXIT_FAILURE);
        }

        GLFWwindow *window = glfwCreateWindow(620, 620,
"Rorate Cube", NULL, NULL);

        if (!window) {
            glfwTerminate();
            exit(EXIT_FAILURE);
        }
        glfwSetMouseButtonCallback(window,
mouseButtonCallback);
        glfwMakeContextCurrent(window);
```

```
        glfwSwapInterval(1);

        glEnable(GL_DEPTH_TEST);

        while (!glfwWindowShouldClose(window))
        {
            float ratio;
            int width, height;
            glfwGetFramebufferSize(window, &width,
&height);
            ratio = width / (float) height;
            glViewport(0, 0, width, height);

            glfwSetKeyCallback(window,
keyboard_callback);

            glClearColor(0, 0, 0, 0);
       glClear(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);

            glLoadIdentity();
            glOrtho(-ratio, ratio, -1.f, 1.f, 1.f,
-1.f);

            glTranslatef(0.4, 0.4, 0.4);
            glTranslatef(trans_x, trans_y, trans_z);

            glScalef(sc_x, sc_y, sc_z);

            glRotatef(rotate_x, 1.0, 0.0, 0.0);
```

```
            glRotatef(rotate_y, 0.0, 1.0, 0.0);
            glRotatef(rotate_z, 0.0, 0.0, 1.0);
            glMultMatrixf(m);
            drawCube();

            glLoadIdentity();
            glOrtho(-ratio, ratio, -1.f, 1.f, 1.f,
-1.f);
            glTranslatef(-0.6, -0.4, -0.6);

            glScalef(0.2, 0.2, 0.2);

            glMultMatrixf(m);
            drawCube();
            glfwSwapBuffers(window);
            glfwPollEvents();
        }
        glfwTerminate();
        return 0;
    }
```

Результат изображен на рис. 2

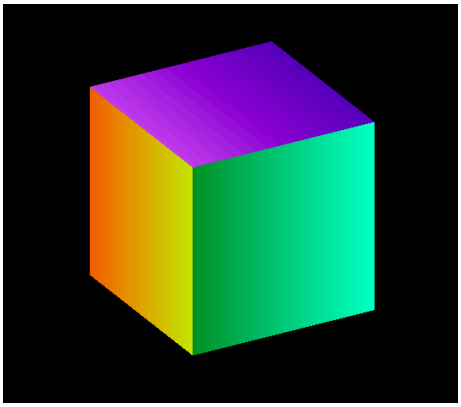

Рисунок 2. Результат работы программы

3. Изменение ориентации и размеров объекта с помощью модельно-видовых преобразований

```
int rotate_y = 0;
int rotate_x = 0;
int rotate_z = 0;

float trans_x = 0;
float trans_y = 0;
float trans_z = 0;

float sc_x = 0.5;
float sc_y = 0.5;
float sc_z = 0.5;

void keyboard_callback GLFWwindow* window, int key,
int scancode, int action, int mods) {

        if (key == GLFW_KEY_ESCAPE && action ==
GLFW_PRESS)
            glfwSetWindowShouldClose(window, GL_TRUE);

        //повороты по x
        if (key == GLFW_KEY_UP && action == GLFW_PRESS)
            rotate_x += 5;
        if (key == GLFW_KEY_DOWN && action ==
GLFW_PRESS)
            rotate_x -= 5;
        //повороты по y
        if (key == GLFW_KEY_RIGHT && action ==
GLFW_PRESS)
```

```
            rotate_y -= 5;
        if (key == GLFW_KEY_LEFT && action ==
GLFW_PRESS)
            rotate_y += 5;
        //повороты по z
        if (key == GLFW_KEY_O && action == GLFW_PRESS)
            rotate_z -= 5;
        if (key == GLFW_KEY_L && action == GLFW_PRESS)
            rotate_z += 5;

        //смещение вверх
        if (key == GLFW_KEY_W && action == GLFW_PRESS)
            trans_y += 0.05;
        //смещение влево
        if (key == GLFW_KEY_A && action == GLFW_PRESS)
            trans_x -= 0.05;
        //смещение вправо
        if (key == GLFW_KEY_D && action == GLFW_PRESS)
            trans_x += 0.05;
        //смещение вниз
        if (key == GLFW_KEY_S && action == GLFW_PRESS)
            trans_y -= 0.05;
    }

    void mouseButtonCallback( GLFWwindow *window, int
button, int action, int mods) {
        //увеличение размера
        if (button == GLFW_MOUSE_BUTTON_RIGHT && action
== GLFW_PRESS) {
            sc_x += 0.2;
```

```
            sc_y += 0.2;
            sc_z += 0.2;
        }
        //уменьшение размера
        if (button == GLFW_MOUSE_BUTTON_LEFT && action
== GLFW_PRESS) {
            if (sc_x > 0.2 && sc_y > 0.2 && sc_z > 0.2)
{
                sc_x -= 0.2;
                sc_y -= 0.2;
                sc_z -= 0.2;
            }
        }
    }
```

4. Реализация возможности переключения между каркасным и твердотельным отображением модели.

```
        //каркасное отображение модели
        if (key == GLFW_KEY_Z && action == GLFW_PRESS)
            glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
        //твердотельное отображение модели
        if (key == GLFW_KEY_X && action == GLFW_PRESS)
            glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
```

5. Результаты выполнения лабораторной работы

Горячие клавиши:

- Esc - закрыть окно;
- A - сдвинуть влево;
- D - сдвинуть вправо;
- W - сдвинуть вверх;
- S - сдвинуть вниз;
- стрелки - вращение фигуры;
- O - вращение фигуры по OZ;
- L - вращение фигуры по OZ;
- Z - каркасное отображение модели;
- X - твердотельное отображение модели;

Управление мышью:

- ПКМ - увеличить фигуру;
- ЛКМ - уменьшить фигуру;

Пример работы программы при каркасном отображении моделей с применением диметрии (рис. 3)

Вывод: В ходе выполнения лабораторной работы были изучены особенности построения трехмерных объектов, функции геометрических преобразований, видовое преобразование, диетическая проекция.

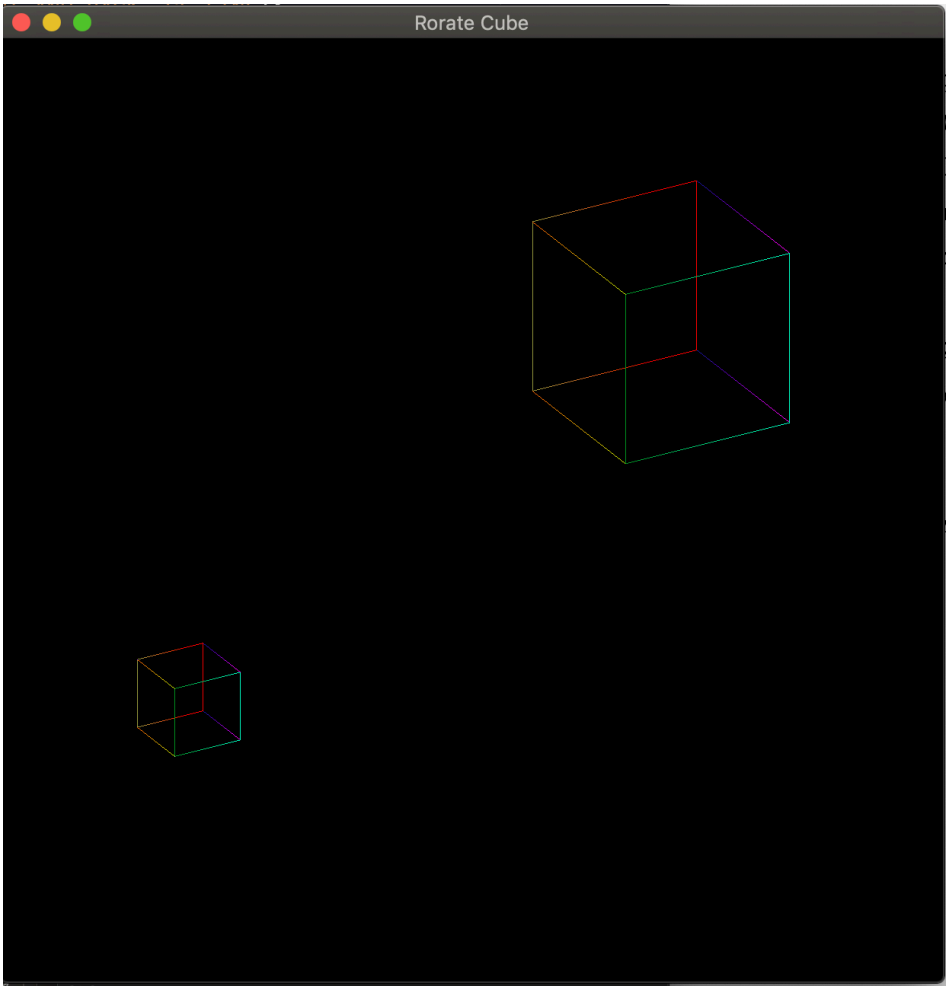

Рисунок 3. Демонстрация работы программы