ГУАП

КАФЕДРА № 42

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

доцент		А.В. Аграновский
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4

Исследование проективных преобразований

по курсу: КОМПЬЮТЕРНАЯ ГРАФИКА

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № 4321		К. А. Лебедев
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

СОДЕРЖАНИЕ

1 Цель работы

Изучение теоретических основ линейных проективных преобразований в трехмерном пространстве. Исследование особенностей практической реализации трехмерных проективных преобразований.

2 Индивидуальное задание

Написать программу на любом языке высокого уровня, реализующую получение анимированной диметрической проекции треугольной призмы, ось вращения которого не параллельна координатным осям.

3 Теоретические сведения

3.1 Базовые аффинные преобразования в трехмерном пространстве

Аффинные преобразования в трехмерном пространстве оказываются более сложными, чем на плоскости. Также усложняется визуализация преобразований на экране компьютера, поскольку для этого требуется получить двумерное изображение трехмерной модели.

Как и в двумерном случае любое аффинное преобразование в трехмерном пространстве может быть представлено в виде суперпозиции вращений, растяжений, отражений и переносов. Обобщим рассмотренные ранее материалы раздела 1 на случай трехмерных преобразований.

Точка в трехмерном пространстве $(x\ y\ z)$ с использованием нормированных однородных координат представляется как [x y z 1]. Аффинное преобразование такой точки описывается выражением

$$[x'y'z'1] = [x\ y\ z1] \times \mathbf{T},$$

где матрица преобразования **T**, в отличии от двумерного случая, имеет размерность 4×4 и в общем виде представляется как:

$$\mathbf{T} = \begin{bmatrix} a & b & c & 0 \\ d & e & f & 0 \\ h & i & j & 0 \\ l & m & n & 1 \end{bmatrix}.$$

Следующие элементы матрицы

$$\begin{bmatrix} a & b & c \\ d & e & f \\ h & i & j \end{bmatrix}.$$

отвечают за покоординатное масштабирование и вращение, а элементы $[l\ m\ n]$ за перенос.

Рассмотрим особенности матрицы **T** для элементарных аффинных преобразований.

1. Матрицы вращения в пространстве:

Матрица вращения вокруг оси абсцисс на угол φ:

$$\mathbf{R_x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & sin\varphi & 0 \\ 0 & -sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Матрица вращения вокруг оси ординат на угол ψ:

$$\mathbf{R_y} = \begin{bmatrix} \cos\psi & 0 & -\sin\psi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\psi & 0 & \cos\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Матрица вращения вокруг оси аппликат на угол θ:

$$\mathbf{R_z} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

За положительное принимается направление против часовой стрелки, если смотреть вдоль оси вращения в сторону начала координат.

2. Матрица растяжения (сжатия), где элементы a, e, $j > 0$ представляют собой коэффициенты растяжения (сжатия) по оси абсцисс, ординат и аппликат соответственно:

$$\begin{bmatrix} a & 0 & 0 & 0 \\ 0 & e & 0 & 0 \\ 0 & 0 & j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3. Матрица отражения.

Матрица отражения относительно плоскости *XOY*

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Матрица отражения относительно плоскости *YOZ*

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Матрица отражения относительно плоскости *ZOX*

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4. Матрица переноса на вектор [l m n]

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ l & m & n & 1 \end{bmatrix}.$$

Для выполнения сложного аффинного преобразования его представляют в виде последовательности базовых преобразований, как и в двумерном случае. При этом следует учитывать, что базовые трехмерные аффинные преобразования определены относительно начала координат, координатных осей или координатных плоскостей. Вычисления при сложном аффинном преобразовании могут быть выполнены либо последовательно, либо путем предварительного определения результирующей матрицы преобразований.

3.2 Особенности представления пространственных многогранников в трехмерном пространстве

Аффинные преобразования выполняются над некоторыми трехмерными фигурами. Если такая фигура представляет собой многогранник, то для программной реализации преобразований проще всего задать ее каркас в виде матрицы однородных координат вершин и массива ребер (в виде пар номеров вершин, соединяемых этими ребрами). Например, единичный куб, одной из вершин которого является начало координат, может быть задан матрицей нормированных однородных координат

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Прорисовка данной фигуры определяется массивом ребер, в котором указываются все пары соединяемых ребрами точек (указанные в массиве номера соответствуют порядковому номеру координат соответствующей точки в матрице однородных координат многоугольника)

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 & 6 & 7 & 8 & 5 & 5 & 6 & 7 & 8 \end{bmatrix}^T$$

В данном учебно-методическом пособии для упрощения формирования исходной трехмерной фигуры в некоторых заданиях мы воспользуемся так называемыми платоновыми телами – правильными многогранниками, состоящими из одинаковых правильных многоугольников и обладающими пространственной симметрией. Правильный многогранник обладает следующими особенностями:

1. Является выпуклым, т. е. расположен по одну сторону плоскости каждого плоского многоугольника на его поверхности;

Таблица 1 – Свойства платоновых тел

Название многогранника	**Число вершин**	**Число ребер**	**Число граней**	**Число сторон у грани**	**Число ребер, примыкающих к вершине**
Тетраэдр	4	6	4	3	3
Октаэдр	6	12	8	3	4
Гексаэдр	8	12	6	4	3
Икосаэдр	12	30	20	3	5

Додекаэдр	20	30	12	5	3

2. Все грани одинаковы и являются правильными многоугольниками;

3. В каждой вершине сходится одинаковое число ребер.

Число вершин (В), граней (Г) и рёбер (Р) любого платонова тела связано формулой Эйлера

$$В + Г = Р + 2.$$

Доказано, что в трехмерном пространстве существует всего 5 платоновых тел, названия которых определяются числом граней: тетраэдр, гексаэдр (куб), октаэдр, додекаэдр и икосаэдр (табл. 1).

3.3 Порядок определения матрицы сложного аффинного преобразования в пространстве

Рассмотрим пример определения матрицы сложного аффинного преобразования в пространстве. Пусть требуется повернуть объект на угол φ относительно прямой *L*, проходящей через точку А с координатами (*l, m, n*) и имеющей направляющий единичный вектор (*u, v, w*).

Учитывая особенности базовых аффинных преобразований, для решения задачи необходимо выполнить следующую последовательность действий:

1. Перенести точку А в начало координат, чтобы прямая *L* проходила через начало координат.

2. Совместить прямую *L* с одной из координатных осей, например, с осью *Z*, для чего

– Повернуть прямую *L* вокруг *OX*

– Повернуть прямую *L* вокруг *OY*

3. Выполнить поворот объекта вокруг прямой *L*.

4. Выполнить преобразования п.2 в обратной последовательности и на углы противоположного знака.

5. Перенести точку А в исходное положение.

Матрица переноса точки А в начало координат

$$\mathbf{T_1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -l & -m & -n & 1 \end{bmatrix}.$$

Для совмещения прямой *L* с осью *Z* необходимо определить углы ψ и θ (рис. 2)

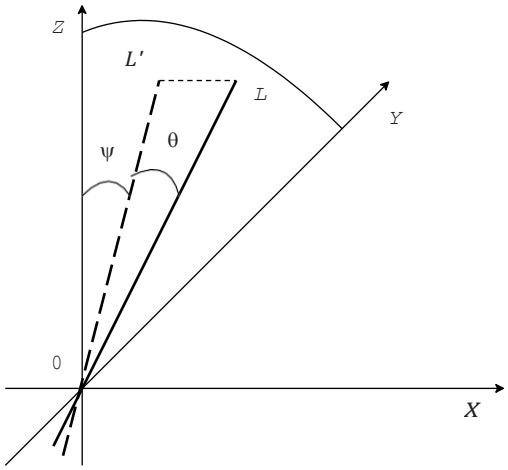

Рис. 1. Определение углов для совмещения прямой L с осью Z

Для нахождения угла ψ рассмотрим прямоугольную проекцию *L'* прямой *L* на плоскость *YOZ*. Направляющий вектор прямой *L'* легко определяется из направляющего вектора прямой *L* и равен (0, *v*, *w*). Тогда

$$\cos\psi = \frac{w}{d}, sin\,\psi = \frac{v}{d},$$

$$d = \sqrt{v^2 + w^2}.$$

При этом матрица вращения вокруг OX на угол ψ имеет вид:

$$\mathbf{R_x} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{w}{d} & \frac{v}{d} & 0 \\ 0 & -\frac{v}{d} & \frac{w}{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

После поворота на угол ψ направляющий вектор прямой L изменится:

$$(u\ \ v\ \ w\ \ 0) \times \mathbf{R_x} = (u\ \ 0\ \ d\ \ 0).$$

Теперь необходимо повернуть прямую *L* вокруг *OY* на угол *θ*, тригонометрические функции которого с учетом начального условия единичности направляющего вектора

$$\sqrt{u^2 + v^2 + w^2} = 1$$

можно определить по формулам

$$cos\theta = u,$$

$$sin\theta = d.$$

Тогда матрица вращения вокруг *OY* на угол θ имеет вид:

$$\mathbf{R_y} = \begin{bmatrix} u & 0 & d & 0 \\ 0 & 1 & 0 & 0 \\ -d & 0 & u & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Поскольку после двух рассмотренных выше поворотов прямая L совмещена с осью Z, можно воспользоваться матрицей вращения объекта на угол φ относительно оси аппликат:

$$\mathbf{R_z} = \begin{bmatrix} cos\varphi & sin\varphi & 0 & 0 \\ -sin\varphi & cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Далее следует вернуться к исходной системе координат, выполнив поворот вокруг OY на угол –θ, затем поворот вокруг OX на угол –ψ и, наконец, перенос на вектор [l m n] с использованием соответствующих обратных матриц.

Результирующая матрица рассмотренного сложного аффинного преобразования будет иметь вид:

$$\mathbf{T} = \mathbf{T_1} \times \mathbf{R_x} \times \mathbf{R_y} \times \mathbf{R_z} \times \mathbf{R_y^{-1}} \times \mathbf{R_x^{-1}} \times \mathbf{T_1^{-1}}.$$

Отметим, что для демонстрации результата трехмерных аффинных преобразований необходимо представить его в экранных координатах. Как правило, начало экранных координат O расположено в левом верхнем углу экрана, ось OX проходит горизонтально слева направо, ось OY – сверху вниз, а ось OZ (при необходимости учета третьей координаты) направлена перпендикулярно экрану в сторону наблюдателя (зрителя). Если объект был сформирован в другой системе координат, может потребоваться переход к экранным координатам, выполняемый с помощью тех же аффинных преобразований.

Следующим необходимым преобразованием для представления результата является проецирование трехмерного объекта на плоскость экрана. Если в дальнейшем не планируется выполнение ка- ких-либо иных действий (например, удаления невидимых граней, формирования сцены, состоящей из нескольких объектов и т. п.), можно ограничиться умножением матрицы нормированных однородных координат получившейся трехмерной фигуры на матрицу ортографической проекции

$$\mathbf{P_{z=0}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

На последнем этапе мы должны перевести полученные координаты каждой точки проекции (x y 0 1) в пиксели на экране с учетом его разрешения.

4 Алгоритм создания аффинных преобразований в пространстве

Для реализации анимированной диметрической проекции треугольной призмы потребуется использовать Canvas API для работы с графикой и анимацией.

4.1. Определение структуры треугольной призмы

Задаются координаты вершин треугольной призмы. Вершины описываются в трехмерном пространстве и могут быть представлены как массив объектов с координатами x, y и z.

Указываются рёбра призмы, соединяющие эти вершины.

4.2. Подготовка оси вращения

Определяется ось вращения, которая не параллельна координатным осям, например, с компонентами (1, 1, 0).

4.3. Проекция на 2D плоскость

Реализуется функция, которая производит проекцию вершин призмы на двумерную плоскость. Проекция осуществляется с учетом глубины (координаты z) для получения корректного визуального восприятия.

4.4. Вращение вокруг произвольной оси

Создается функция, вычисляющая новые координаты вершин после их вращения вокруг заданной оси на определенный угол. Это достигается с помощью матриц вращения.

4.5. Отрисовка призмы

Определяется функция для очистки текущего кадра и отрисовки обновленных вершин и рёбер призмы. Используются функции для рисования линий, соединяющих вершины.

4.6. Создание анимации

Анимация запускается с помощью цикла, который обновляет угол поворота и перерисовывает призму. Используется метод для плавного обновления кадра.

4.7. Запуск анимации

Анимация инициируется, и результат отображается в отдельном окне.

5 Язык программирования и используемые библиотеки

Для реализации программы построения фигур и аффинных преобразований был использован язык программирования JavaScript, а также следующие API.

5.1 Язык программирования JavaScript

JavaScript был выбран в качестве основного языка программирования для реализации задачи ввиду его расширяемости, читабельности и мультипарадигменности его предметной области. Браузерный Javascript поддерживает однопоточность и предоставляет API для работы с графикой в браузере (Canvas API), что делает его удобным инструментом для построения сложных математических структур, таких как проективные преобразования.

5.2 Используемые API

Canvas API — это API для двумерного рисования. Он позволяет рисовать линии, фигуры, изображения и текст прямо в браузере без использования плагинов, таких как Flash или Java. Canvas изначально был создан Apple для своих виджетов, но с тех пор был принят всеми разработчиками основных браузеров и теперь является частью спецификации HTML5.

Browser API — это интерфейс прикладного программирования для веб-сервера или веб-браузера. Это концепция веб-разработки, обычно ограниченная клиентской стороной веб-приложения (включая любые используемые веб-фреймворки), и поэтому обычно не включает детали реализации веб-сервера или браузера, такие как SAPI или API, если они не доступны для общего доступа через удаленное веб-приложение.

5.3 Используемые интерфейсы

document: используется для манипуляций с DOM (древовидная структура элементов веб страницы).

requestAnimationFrame: указывает браузеру на то, что вы хотите произвести анимацию, и просит его запланировать перерисовку на следующем кадре анимации. В качестве параметра метод получает функцию, которая будет вызвана перед перерисовкой.

6 Описание программы

1) Определение координат вершин призмы

Программа начинается с определения координат шести вершин треугольной призмы в трехмерном пространстве. Каждая вершина представлена как объект с полями x, y и z.

2) Задание оси вращения

Ось вращения задается как вектор { x: 1, y: 1, z: 0 }. Этот вектор указывает направление вращения в пространстве и проходит через одну из вершин призмы.

3) Инициализация угла вращения

Переменная angle инициализируется значением 0. Она используется для отслеживания текущего угла вращения призмы в процессе анимации.

4) Функция проекции

Функция project проецирует 3D-координаты вершин призмы на 2D-плоскость. Проекция учитывает перспективу, изменяя размеры вершин в зависимости от их координаты z, что позволяет корректно визуализировать 3D-объект на двумерном экране.

5) Функция вращения

Функция rotate осуществляет вращение вершины призмы вокруг заданной оси. Для этого используются матрицы вращения, которые вычисляют новые координаты x, y и z. Функция принимает вершину и угол вращения в качестве аргументов и возвращает новые координаты.

6) Функция отрисовки

Функция draw отвечает за отрисовку призмы на канвасе. Она очищает предыдущие кадры, применяет вращение и проекцию ко всем вершинам, а затем рисует контуры призмы, соединяя вершины линиями. В конце каждой итерации увеличивается угол вращения, создавая эффект анимации.

7) Запуск анимации

Анимация запускается с вызова функции draw, что создает бесконечный цикл рисования и вращения призмы, придавая ей динамичность и эффект движения.

7 Результаты работы программы

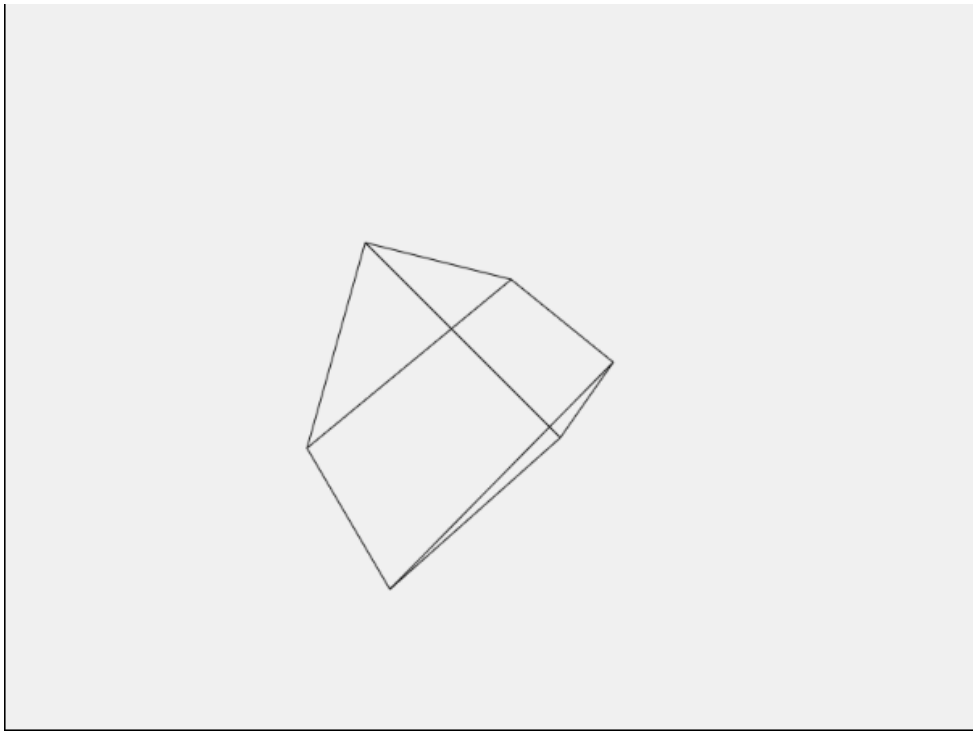

Рис. 2 – Результат работы программы №1.

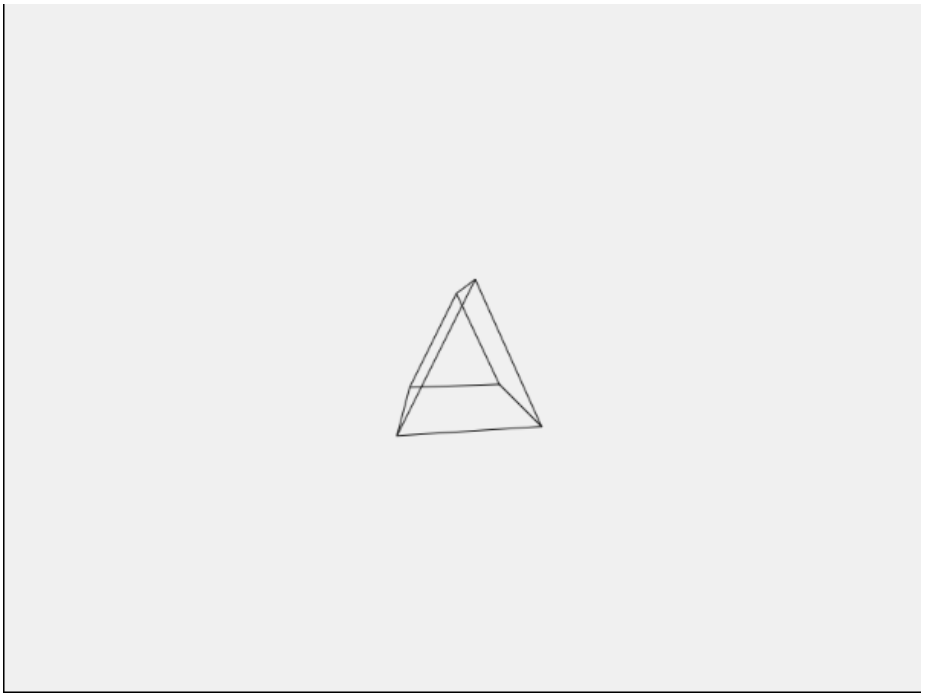

Рис 3 – Результат работы программы №2.

Вывод

В процессе выполнения этой работы были получены следующие теоретические знания и навыки:

1) Модульная структура: Разработана четкая структура программы, что обеспечивает хорошую организацию кода и его повторное использование.

2) Матричные операции: Реализованы основные матричные операции для аффинных преобразований, включая вращение и масштабирование.

3) Интуитивный интерфейс: Создан визуальный интерфейс для демонстрации процесса деформации треугольника, позволяющий пользователю легко наблюдать изменения.

4) Плавная анимация: Достигнута плавность анимации благодаря эффективному использованию функции requestAnimationFrame.

5) Освоены аффинные преобразования и их матричное представление, что позволило лучше понять их применение в графике.

6) Получен опыт работы с графическими элементами в JavaScript, что расширяет навыки веб-разработки.

7) Навык по созданию анимации, применяя принципы проективных преобразований и физику движения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГУАП, документация для учебного процесса. – URL:https://guap.ru/regdocs/docs/uch (дата обращения 10.05.2024)
2. Руководство по JavaScript – URL: https://developer.mozilla.org/ru/docs/Web/JavaScript
3. Мартин Роберт, Чистый код: создание, анализ и рефакторинг / Мартин Роберт. — СПб.: Питер, 2022. — 464 с.
4. 5) HTML5: Canvas. Основы– URL: https://webmaster.alexanderklimov.ru/html/canvas_basic.php
5. Линейная алгебра: учебник и практикум для вузов / Е. Б. Бурмистрова, С. Г. Лобанов — Москва.: Издательство Юрайт, 2024. — 421 с.
6. Использование методов преобразования координат для формирования растровых изображений: учеб.-метод. пособие / А. В. Аграновский. – СПб.: ГУАП, 2024. – 40 с.
7. Основы компьютерной графики: учебник и практикум для вузов/ А. В. Боресков, Е. В. Шикин — Москва.: Издательство Юрайт, 2024. — 219 с

ПРИЛОЖЕНИЕ А

```
const canvas = document.getElementById("canvas");
const ctx = canvas.getContext("2d");

// Определяем координаты вершин треугольной призмы
const vertices = [
    { x: -50, y: -50, z: -50 },
    { x: 50, y: -50, z: -50 },
    { x: 0, y: 50, z: -50 },
    { x: -50, y: -50, z: 50 },
    { x: 50, y: -50, z: 50 },
    { x: 0, y: 50, z: 50 },
];

const rotationAxis = { x: 1, y: 1, z: 0 }; // Ось вращения
let angle = 0;

function project(vertex) {
    // Проекция на 2D плоскость
    const scale = 200 / (200 + vertex.z);
    const x = vertex.x * scale + canvas.width / 2;
    const y = -vertex.y * scale + canvas.height / 2;
    return { x, y };
}

function rotate(vertex, angle) {
    // Вращение вокруг произвольной оси
    const cos = Math.cos(angle);
    const sin = Math.sin(angle);
    const { x, y, z } = vertex;

    // Применяем матрицы вращения
    const newX =
        (cos + (1 - cos) * rotationAxis.x * rotationAxis.x) * x +
        ((1 - cos) * rotationAxis.x * rotationAxis.y - rotationAxis.z *
sin) *
            y +
        ((1 - cos) * rotationAxis.x * rotationAxis.z + rotationAxis.y *
sin) *
            z;

    const newY =
        ((1 - cos) * rotationAxis.y * rotationAxis.x + rotationAxis.z *
sin) *
            x +
```

```
        (cos + (1 - cos) * rotationAxis.y * rotationAxis.y) * y +
        ((1 - cos) * rotationAxis.y * rotationAxis.z - rotationAxis.x *
sin) *
            z;

    const newZ =
        ((1 - cos) * rotationAxis.z * rotationAxis.x - rotationAxis.y *
sin) *
            x +
        ((1 - cos) * rotationAxis.z * rotationAxis.y + rotationAxis.x *
sin) *
            y +
        (cos + (1 - cos) * rotationAxis.z * rotationAxis.z) * z;

    return { x: newX, y: newY, z: newZ };
}

function draw() {
    ctx.clearRect(0, 0, canvas.width, canvas.height);

    // Применяем вращение и проекцию к каждой вершине
    const projectedVertices = vertices.map(vertex => {
        const rotated = rotate(vertex, angle);
        return project(rotated);
    });

    // Рисуем контуры призмы
    ctx.beginPath();
    for (let i = 0; i < 3; i++) {
        ctx.moveTo(projectedVertices[i].x, projectedVertices[i].y);
        ctx.lineTo(
            projectedVertices[(i + 1) % 3].x,
            projectedVertices[(i + 1) % 3].y
        );
        ctx.moveTo(projectedVertices[i + 3].x, projectedVertices[i +
3].y);
        ctx.lineTo(
            projectedVertices[((i + 1) % 3) + 3].x,
            projectedVertices[((i + 1) % 3) + 3].y
        );
    }
    for (let i = 0; i < 3; i++) {
        ctx.moveTo(projectedVertices[i].x, projectedVertices[i].y);
        ctx.lineTo(projectedVertices[i + 3].x, projectedVertices[i +
3].y);
    }
```

```
    ctx.strokeStyle = "black";
    ctx.stroke();

    angle += 0.01; // Увеличиваем угол для анимации
    requestAnimationFrame(draw);
}

draw(); // Запуск анимации
```

Листинг Программы