Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет систем управления и робототехники

Лабораторная работа № 3
"Матрицы в 3D-графике "

по дисциплине Практическая линейная алгебра

Выполнила: студентка гр. **R3238**

Нечаева А. А.

Преподаватель: *Перегудин Алексей Алексеевич*

Санкт-Петербург, 2023-2024

1 Задание. Создайте кубик.

1.1 Как работает код?

В первой части кода (рисунок 1) задаются координаты вершин куба: каждый столбец – вершина и сверху вниз в нем заданы координаты x, y, z в пространстве и w (последняя отвечает за перспективу).

```
verticesCube = [
    -1, 1, 1,-1,-1, 1, 1,-1;
    -1,-1, 1, 1,-1,-1, 1, 1;
    -1,-1,-1,-1, 1, 1, 1, 1;
     1, 1, 1, 1, 1, 1, 1, 1
    ];

```

Рис. 1. Исходный код кубика, часть 1.

Вторая часть (рисунок 2) отвечает за задание плоскостей граней куба, в каждой строчке записаны 4 вершины куба, по которым строится грань.

```
facesCube = [
          1, 2, 6, 5;
          2, 3, 7, 6;
          3, 4, 8, 7;
          4, 1, 5, 8;
          1, 2, 3, 4;
          5, 6, 7, 8
         ];
```

Рис. 2. Исходный код кубика, часть 2.

Функция *DrawShape* отвечает за отрисовку кубика, сначала строятся точки вершин по 3 координатам и с учетом перспективы, затем изображаются грани.

```
16
17 DrawShape(verticesCube, facesCube, 'blue')
18 axis equal;
19 view(3);
20
21 function DrawShape(vertices, faces, color)
22     patch('Vertices', (vertices(1:3,:)./vertices(4,:))', 'Faces', faces, '
       FaceColor', color);
23 end
```

Рис. 3. Исходный код кубика, часть 3.

1.2 Почему используется четырехкомпонентный вектор, а не трех?

Четвертый компонент в векторе позволяет реализовывать перспективную проекцию, а не только отображать ортогональную проекцию. Кроме того, с помощью матрицы 4×4 реализуются такие преобразования как сдвиги, повороты и т.д. в трехмерном пространстве.

1.3 Как задать другие фигуры?

2 Задание. Изменить масштаб кубика.

Для изменения масштаба кубика использовалась матрица вида:

$$\begin{bmatrix} a_1 & 0 & 0 & 0 \\ 0 & a_2 & 0 & 0 \\ 0 & 0 & a_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

Где, a_1, a_2, a_3 отвечают за изменение масштаба по x, y и z соответственно.

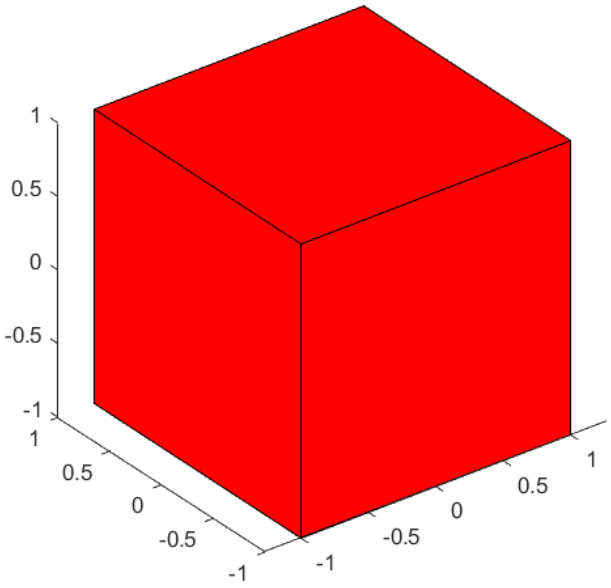

Рис. 4. Оригинальный масштаб,при $a_i = 1$.

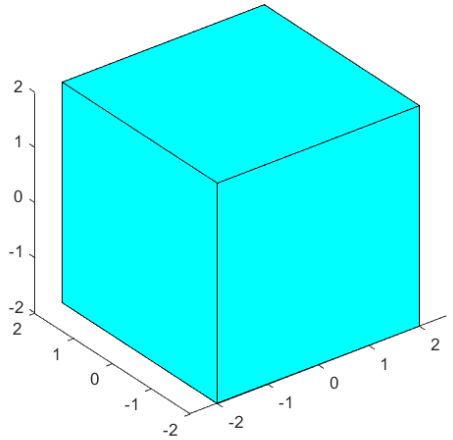

Рис. 5. Результат при $a_i = 2$.

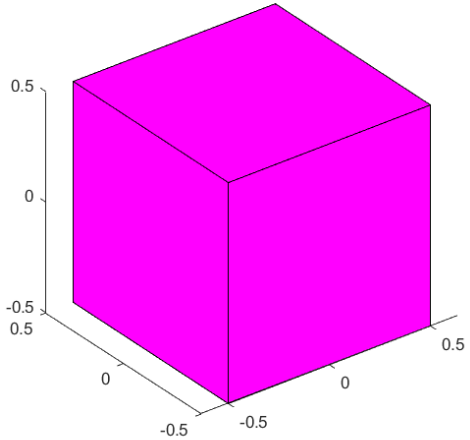

Рис. 6. Результат при $a_i = 0.5$.

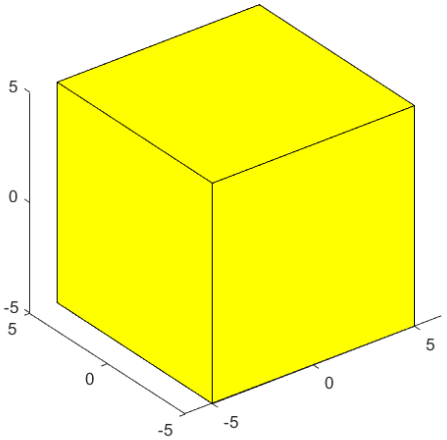

Рис. 7. Результат при $a_i = 5$.

```
sizeMatrix = [
    5, 0, 0, 0;
    0, 5, 0, 0;
    0, 0, 5, 0;
    0, 0, 0, 1
    ];

newVertices = sizeMatrix * verticesCube;

DrawShape (newVertices, facesCube, 'y')
```

Листинг 1. Часть кода, отвечающая за масшабирование кубика.

3 Задание. Переместить кубик.

4 Задание. Выполнить вращение кубика.

5 Задание. Выполнить вращение кубика около одной вершины.

6 Задание. Реализация камеры.

7 Задание. Реализация перспективы.

8 Задание. * Почти Minecraft.

Для визуализации был написан код на языке *Python* с использованием библиотек *Matplotlib* и *Numpy*.
Код расположен на **GitHub**.
Отражение (симметрию) плоскости относительно прямой $y = ax$, *в нашем случае после подстановки* $a = 2$, *получаем* $y = 2x$. *Задача – найти матрицу вида*: