В этой лабораторной вы познакомитесь с особыми матрицами преобразования пространства, которые главным образом используются в 3D-графике. Перед выполнением задания разберитесь с тем, как они устроены. Матрицы преобразования пространства. Матрицы преобразования перспективы (Во втором источнике матрицы транспонированы). Все преобразования координат в данной работе должны быть осуществлены с помощью матриц!

Задание 1. Создайте кубик. Используйте код MATLAB (1) или Python (2) для отрисовки кубика в 3D пространстве. Разберитесь, как работает выбранный вами код. Почему мы используем четырехкомпонентный вектор, а не трех? Как задать другие фигуры?

Задание 2. Измените масштаб кубика. Придумайте матрицу преобразования масштаба по осям S исходя из изученных материалов. Примените эту матрицу ко всем векторам кубика, таким образом получив его образ. Продемонстрируйте, как меняется кубик в зависимости от выбранной матрицы масштабирования.

Задание 3. Переместите кубик. Придумайте матрицу перемещения T. Разберитесь, как она работает. Исследуйте, как меняется кубик в зависимости от выбранной комбинации матриц перемещения и масштабирования TS и ST. Эквиваленты ли такие преобразования?

Задание 4. Выполните вращение кубика. Придумайте вектор v, вокруг которого вы хотите вращать кубик на угол θ. Используйте формулу

$$v = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}, \quad J = \frac{1}{\|v\|} \begin{bmatrix} 0 & -v_z & v_y & 0 \\ v_z & 0 & -v_x & 0 \\ -v_y & v_x & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad R_v(\theta) = e^{J\theta}$$

для получения матрицы поворота $R_v(\theta)$ и примените её к кубику. Исследуйте, как меняется кубик в зависимости от выбранных оси и угла. Разберитесь, как представленная формула работает. Какие матричные функции $R_x(\theta)$, $R_y(\phi)$, $R_z(\psi)$ мы получим, если выберем вектор v вдоль осей x, y или z соответственно? Является ли тройка матриц $R_x(\theta)R_y(\phi)R_z(\psi)$ достаточной для описания всех возможных вращений в 3D пространстве? Можно ли восстановить ось вращения? (см. Теорема вращения Эйлера). На графики добавьте вектор, коллинеарный оси вращения.

Задание 5. Выполните вращение кубика около одной вершины. Найдите матрицу преобразования, которая осуществит вращение кубика таким образом, что центр вращения будет совпадать с одной из его вершин. Продемонстрируйте результат, разместив на одном графике исходный и повернутый кубик (Результат работы должен быть виден).

Задание 6. Реализация камеры. Разместите нескольких разных кубиков на одном графике, таким образом создав сцену. Используйте команду MATLAB `view([0 -90])` или Python `ax.view_init(azim=0, elev=-90)`, чтобы посмотреть на график "снизу" (1). Выберете матрицы T_c и $R_c(\theta)$, отвечающие за положение и поворот камеры так, чтобы она смотрела на сцену на отдалении под углом. Найдите такую матрицу C^{-1}, которая реализует обратное преобразование положения камеры, переместив её из выбранного положения в начало координат с стандартным поворотом. Примените эту матрицу ко всем объектам сцены. Какой эффект мы таким образом получим?

Задание 7. Реализация перспективы. Используйте матрицу перспективы P из изученных материалов для преобразования сцены из задания 6. Как эта матрица устроена? Какие параметры нам необходимо выбрать для её реализации? Используйте стандартное вращение графика с помощью команд из предыдущего задания для исследования деформации пространства под её влиянием.

Задание 8 (необязательное). Почти Blender. Постройте домик или любое другое строение из блоков или других фигур. Используйте матрицы масштабирования, поворота и перемещения, а также задайте камеру и перспективу.

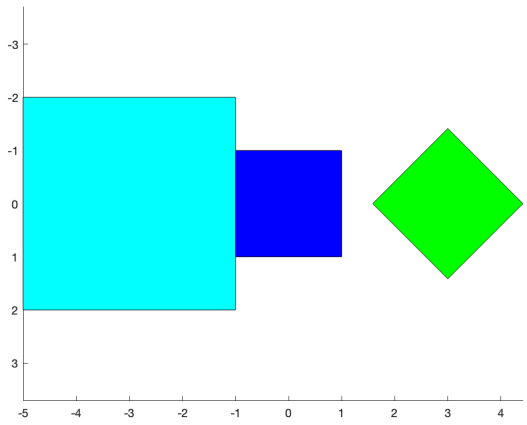

Рис. 1: Сцена из трех кубиков.

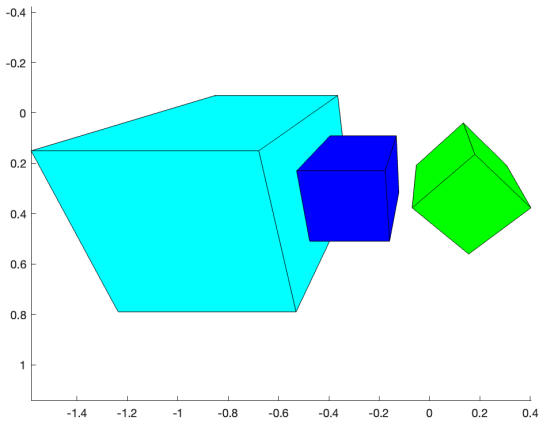

Рис. 2: Результат выполнения задания 7.

```
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
    ];

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
        ];

DrawShape(verticesCube, facesCube, 'blue')
axis equal;
view(3);

function DrawShape(vertices, faces, color)
    patch('Vertices', (vertices(1:3,:)./vertices(4,:))', 'Faces', faces, '
   FaceColor', color);
end
```

Листинг 1: Код для отрисовки кубика на языке MATLAB.

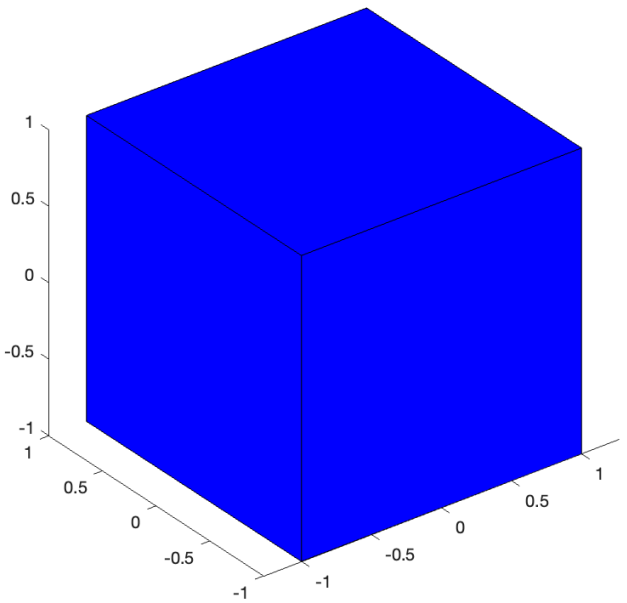

Рис. 3: Результат выполнения программы.

```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.art3d import Poly3DCollection

vertices_cube = np.array([
    [-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]
])

faces_cube = np.array([
    [0, 1, 5, 4],
    [1, 2, 6, 5],
    [2, 3, 7, 6],
    [3, 0, 4, 7],
    [0, 1, 2, 3],
    [4, 5, 6, 7]
])

fig = plt.figure()
ax = fig.add_subplot(projection='3d', proj_type = 'ortho')

def draw_shape(vertices, faces, color):
    vertices = (vertices[:3, :] / vertices[3, :]).T
    ax.add_collection3d(Poly3DCollection(vertices[faces], facecolors=color
    , edgecolors='k', linewidths=0.2))

draw_shape(vertices_cube, faces_cube, 'blue')

ax.set_box_aspect([1,1,1])
ax.set_xlim(-1, 1); ax.set_ylim(-1, 1); ax.set_zlim(-1, 1)
ax.view_init(azim=-37.5, elev=30)
ax.set_xticks(np.linspace(-1, 1, 5)); ax.set_yticks(np.linspace(-1, 1, 5))
    ; ax.set_zticks(np.linspace(-1, 1, 5))

plt.show()
```

Листинг 2: Код для отрисовки кубика на языке Python.

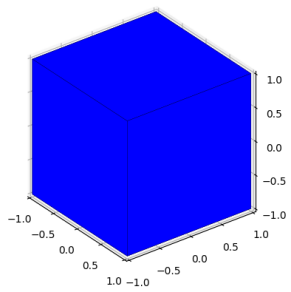

Рис. 4: Результат выполнения программы.